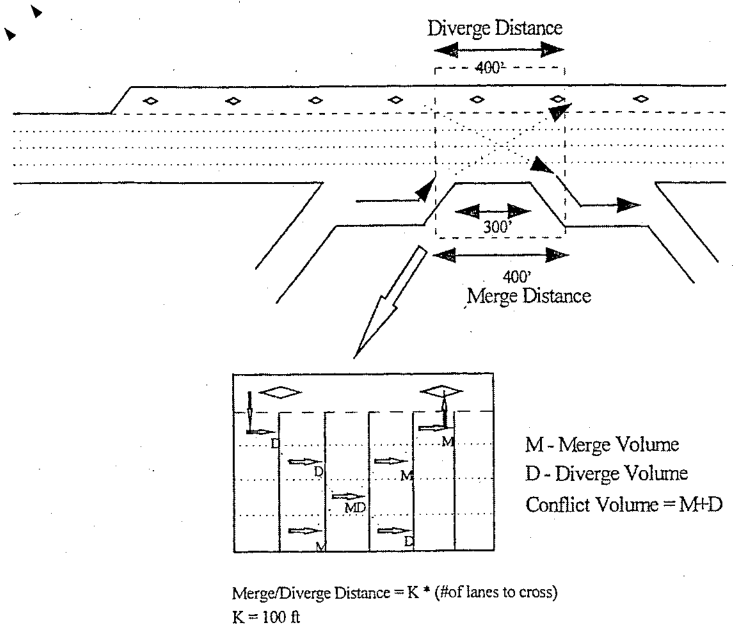
Final Report 1999-11

PB99-157349



Enhancements of the KRONOS Simulation Package and Database for Geometric Design, Planning, Operations, and Traffic Management in Freeway Networks/Corridors (Phase III)

Technical Report Documentation Page

1. Report No. MN/RC - 1999-11	2.	3. Recipient's Accession No.
4. Title and Subtitle ENHANCEMENTS OF THE KRONOS SIMULATION PACKAGE AND DATABASE FOR GEOMETRIC DESIGN PLANNING, OPERATIONS AND TRAFFIC MANAGEMENT IN FREEWAY NETWORKS/CORRIDORS (PHASE III)		5. Report Date July 1997
		6.
7. Author(s) Dr. Eil Kwon, Ramesh Kota, Michael Coyle, Panos Michalopoulos, Sejun Song		8. Performing Organization Report No.
9. Performing Organization Name and Address University of Minnesota, 500 Pillsbury Drive SE, Minneapolis, MN 55455		10. Project/Task/Work Unit No.
		11. Contract (C) or Grant (G) No. (C) 72077 TOC # 139
12. Sponsoring Organization Name and Address Minnesota Department of Transportation, 395 John Ireland Boulevard, St.Paul Minnesota, 55155		13. Type of Report and Period Covered Final Report 1997
		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

This report summarizes the final results of the research effort to develop a freeway traffic simulator with the capability to evaluate freeway operational strategies, such as traffic-responsive ramp metering and high-occupancy vehicles (HOV) lanes.

Researchers first developed an efficient software data structure by adopting a dynamic memory allocation scheme to use the available memory as efficiently as possible. That work also included modifying the existing macroscopic, segment-based modeling structure and developing new types of pipeline segments to facilitate detection modeling and further model enhancements. Based on the new segment-based modeling structure, researchers developed a new simulation module to handle HOV lane traffic flows and extended the simulation procedure for an exclusive HOV lane to handle a network of freeways.

Further, the simulation model also incorporates a new module to emulate the traffic-responsive ramp metering algorithm implemented by the Traffic Management Center since the 1980s. The new software structure developed in this research allows the future addition of new metering algorithms without major difficulties. To facilitate the data input process for the expanded simulation features, a new Windows-based user interface was developed using the Delphi software development tool kit. With the new user interface, most of the data input process can be done without exiting the main menu screen.

17. Document Analysis/Descriptors Traffic modeling, Macroscopic simulation, Freeway merging/diverging, Multi-stage incident, Traffic delay, Interrupted flow		18. Availability Statement No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161	
19. Security Class (this report) Unclassified	20. Security Class (this page) Unclassified	21. No. of Pages 90	22. Price

ENHANCEMENT OF THE KRONOS SIMULATION PACKAGE AND DATABASE FOR GEOMETRIC DESIGN, PLANNING, OPERATIONS AND TRAFFIC MANAGEMENT IN FREEWAY NETWORKS/CORRIDORS (Phase III)

Final Report

Prepared by

Dr. Eil Kwon
Center for Transportation Studies
University of Minnesota
Tel: 612-625-1371
Fax: 612-625-6381

Ramesh Kota, Michael Coyle and Dr. Panos Michalopoulos
Department of Civil Engineering
University of Minnesota

Sejun Song
Department of Computer Science
University of Minnesota

July 1997

Published by

Minnesota Department of Transportation
Office of Research Services
First Floor, 395 John Ireland Boulevard, MS 330
St. Paul, Minnesota 55155

TABLE OF CONTENTS

I. INTRODUCTION

I.1 Background	1
I.2 Research Objective	2
I.3 Report Organization	2

II. DEVELOPMENT OF EFFICIENT SOFTWARE STRUCTURE FOR NETWORK SIMULATION

II.1 Limitations in the current Kronos software structure	3
II.2 Design of new data structure with dynamic memory allocation	4
II.3 Other enhancements in software structure	6

III. ENHANCEMENT OF SEGMENT-BASED FLOW MODELING STRUCTURE

III.1 Limitations in current modeling structure	13
III.2 Development of new pipeline-segment types	15
III.3 Simulation procedure for new pipeline segments	16
III.4 Boundary treatments between two segments with different capacities	19

IV. DEVELOPMENT OF EXCLUSIVE HOV LANE AND FREEWAY NETWORK SIMULATION MODULE

V.1 Overview of HOV lane and network simulation	23
V.2 Development of new freeway data structure for network simulation	23
V.3 Input/output file structure for network simulation	28
V.4 Network simulation process	29
V.5 Testing network simulation module	30

V. MODELING NON-EXCLUSIVE HOV LANE TRAFFIC FLOWS

V.1 Overview of diamond HOV lane flow simulation	33
V.2 Modeling beginning and ending segments for diamond lanes	33
V.3 Merging/diverging between HOV and normal lanes	37
V.4 Testing diamond HOV lane module	41

VI. DEVELOPMENT OF TRAFFIC RESPONSIVE, AUTOMATIC RATE-SELECTION RAMP METERING SIMULATION MODULE

VI.1 Overview of Mn/DOT traffic responsive ramp metering scheme 43
VI.2 Development of a simulation module for Mn/DOT 80 metering algorithm 46
VI.3 Overview of new metering algorithm: Mn/DOT 97 52

VII. DEVELOPMENT OF WINDOWS-BASED USER INTERFACE AND DEMAND DATA LOADING PROCEDURE

VII.1 Overview of Windows-based user interface 55
VII.2 Design and implementation of input/output screens with Delphi 56
VII.3 Development of automatic demand data loading procedure 68

VIII. PRELIMINARY STUDY FOR ESTIMATION OF FLOW-DENSITY RELATIONSHIP

VIII.1 Flow-density relationship in macroscopic simulation 71
VIII.2 Development of new flow-density curve data structure 72
VIII.3 Framework for an automatic calibration for q-k relationships for a given freeway 75

IX. CONCLUSIONS AND FUTURE RESEARCH NEEDS 81

BIBLIOGRAPHY

List of Tables

Table 5-1	Test results for Diamond HOV lane simulation module	42

List of Figures

Figure 3-1	Segment types with lead/lag pipeline in Kronos 8	14
Figure 3-2	New pipeline segment types	15
Figure 4-1	Test results for network simulation module	31
Figure 5-1	Diamond HOV lane starting segment types	34
Figure 5-2	Diamond HOV lane ending segment types	35
Figure 5-3	Merging/Diverging between HOV and normal lanes	39
Figure 5-4	Q-K relationship	39
Figure 5-5	Relationship between conflict index and capacity reduction factor	39
Figure 5-6	Geometrics and traffic demand pattern for diamond lane module testing	42
Figure 7-1	Freeway geometry data input window	57
Figure 7-2	Global flow-density relationship data input window	58
Figure 7-3	Volume/Occupancy threshold file data input window	59
Figure 7-4	Example segment data input window	60
Figure 7-5	Example metering period data input window	61
Figure 7-6	Example spreadsheet output window for total	63
Figure 7-7	Example output contour window	64
Figure 7-7a	Available options for contour output	65
Figure 7-8	Example 2-D output screen	66
Figure 7-8a	Example 3-D output screen	67
Figure 7-9	Structure of automatic demand data loading procedure	69
Figure 8-1	General form of flow-density relationship adopted in Kronos	71
Figure 8-2	Initialization procedure for q-k curve data	73
Figure 8-3	Framework for automatic calibration of q-k relationship	76

ACKNOWLEDGEMENTS

This project was sponsored by the Minnesota Department of Transportation. The Traffic Management Center, Minnesota Department of Transportation, is also acknowledged for its assistance in providing the data necessary for this study. The views expressed here are those of the authors and do not necessarily reflect the views of the sponsor.

EXECUTIVE SUMMARY

One of the key elements in improving freeway operations is the ability to assess the effectiveness of various operational and design alternatives prior to implementation. While simulation methods have long been recognized as the most powerful tool for such analysis, most existing freeway simulation software lack the sophistication necessary for the applications requiring reasonably high performance levels. To resolve the above problems, a personal computer-based, freeway simulation software, called Kronos, has been developed by the research team in the University of Minnesota with the support from the Minnesota Department of Transportation (Mn/DOT). This report summarizes the final results of the current research effort to enhance Kronos as a tool that can be applicable in evaluating freeway operational strategies, such as traffic-responsive ramp metering and High Occupancy Vehicle (HOV) lanes. First, an efficient software data structure was developed by adopting a dynamic memory allocation scheme, so that available memory can be used as efficiently as possible. The existing macroscopic, segment-based modeling structure was also modified and new types of pipeline segments were developed to facilitate detection modeling and further model enhancements. While this work was not included in the original work plan, the inefficiency in the previous modeling structure, where the segment types were redefined internally to confirm the requirements of the simulation module, was discovered during the process of developing the traffic-responsive ramp metering module. To prevent excessive complexity in modeling detectors and to reduce the possibility of the potential error because of the internal type conversion, a new definition of freeway segment types was introduced and a new simulation procedure for each new segment type was developed. The newly defined pipeline segments have different internal flow configurations depending on the types of preceding and following segments. Due to the additional work involved in this modeling structure modification, the testing of the ramp metering module with the real traffic data will be conducted later jointly with the Mn/DOT traffic engineers. Based on the new segment-based modeling structure, a new simulation module to handle HOV lane traffic flows was developed and the simulation procedure to treat an exclusive HOV lane was extended to handle a network of freeways. Further, a new module to emulate the traffic-responsive ramp metering algorithm implemented by the Traffic Management Center since 1980's has been developed and incorporated into the simulation module. The new software structure developed in this research allows the addition of new metering algorithms in the future without major difficulties. To facilitate the data input process for the expanded

simulation features, the enhancement of the existing DOS-based user interface was initially tried. However, it was soon found out that it would be very difficult to modify the existing user interface because of the limitations in the DOS-based programming environment. Further, the recent trend strongly indicates that the Windows operating system has been used as the most popular operating environment for personal computer-based applications. As a result, a new Windows-based user interface was developed using the Delphi software as the development tool kit. The new user interface is a full Windows-based application and most of the data input process could be done without exiting the main menu screen. The preliminary study to automatically estimate the flow-density relationships for a given section of a freeway with real traffic data was performed and a framework for an optimal calibration process for the flow-density relationships was developed using a non-linear optimization routine that did not require the calculation of derivatives.

Future research needs include the continuous testing and debugging of the new software developed from this research. The budget and time limitations in this research did not allow to perform an extensive testing and debugging of the software. Further, the screen design of the new user interface can be improved with the feedback from the traffic engineers. Secondly, the traffic models developed and incorporated into the simulation module need to be continuously tested and enhanced with the real traffic data from various traffic and weather conditions. The flow-density relationships of the various segment types under the different traffic and weather conditions need to be studied and a base set of default relationships can be developed. Finally, an efficient traffic database that can store and manage the traffic data from the existing detectors needs to be developed, so that the user interface can directly extract the traffic data from the database and create the demand data necessary for simulation.

I. INTRODUCTION

I.1 Background

One of the key elements in improving freeway operations is the ability to assess the effectiveness of various operational and design alternatives prior to implementation. While simulation methods have long been recognized as the most powerful tool for such analysis, most existing freeway simulation software lack the sophistication necessary for the applications requiring reasonably high performance levels. Furthermore, there is practically no integration between simulation and data management resulting in the time-consuming manual effort to gather, manipulate and process the data and then analyze the results. In addition, the lack of efficient, user-friendly interface has forced traffic engineers to spend substantial amount of time to learn the data input process, which has made the employment of simulation unattractive.

To resolve the above problems, a personal computer-based, freeway simulation software, called Kronos, has been developed by the research team in the University of Minnesota (Michalopoloulos, et. al. 1993, Kwon, et. al., 1995). The previous phases of this research have developed the traffic models to simulate the traffic behavior in the various freeway segments including multi-stage incidents. A graphical, interactive user interface was also developed under the MS-DOS environment. The user interface has first introduced the concept of the graphical, interactive data-input process to traffic simulation and enabled the user to build a freeway section graphically on the computer screen using the segment icons with a mouse. This research is the final phase of the ongoing effort to develop Kronos as a tool that can be applicable in evaluating traffic operations and management strategies, such as traffic-responsive ramp metering and HOV lanes. Further, there exists a strong need to improve the structure of the existing DOS-based software, whose complexity and the limitations in the DOS operating environment make any addition of new features very difficult. For an effective evaluation of various operational strategies, it is of critical importance to develop a simulation tool with a flexible structure, so that continuous enhancements and interaction with other applications can be performed without major difficulties.

I.2 Research objectives

The major objectives of this research include:

- Development of efficient software structure to handle a large network.
- Development of a Windows-based user interface and traffic demand data loading procedure.
- Development of a new module to simulate the current Mn/DOT real time ramp metering strategy.
- Development of a new simulation module to treat HOV lanes.
- Preliminary study for developing automatic, optimal calibration method for flow-density relationships.

In addition, a new segment modeling structure was developed to address to the problems found during the development and testing of the ramp-metering module. With the new segment modeling structure, the initial segment definition by the user for a given freeway remains same throughout simulation without requiring internal type conversion, which has been done in the current version. While this work was not included in the work plan, it was needed to facilitate the addition of new simulation modules, such as real time detection and traffic-responsive ramp metering. Further, by eliminating the internal segment type conversion with the new modeling structure, the input data file generation process became simpler and more robust than the previous one.

I. 3 Report organization

Chapter II develops an efficient software structure adopting the dynamic memory allocation scheme to handle a large network. The enhancement of the segment modeling structure is included in Chapter III. Chapter IV develops a new simulation module to simulate exclusive HOV lanes and a network of freeways. The modeling of non-exclusive HOV lanes is described in Chapter V. Chapter VI includes the development of the traffic-responsive ramp metering simulation module and Chapter VII summarizes the development of a new Windows-based user interface. The preliminary study for estimating flow-density relationships is included in Chapter VIII and finally Chapter IX summarizes the conclusions and future research needs.

II. DEVELOPMENT OF EFFICIENT SOFTWARE STRUCTURE FOR FREEWAY NETWORK SIMULATION

II. 1 Limitations in the Current Kronos Software Structure

The current structure of the Kronos freeway simulation software, which is MS-DOS based, has the following major limitations:

- Subject to DOS 640 KB barrier,
- Static memory allocation,
- Inefficient data structure with no generic freeway representation.

Further, the user interface has been written with the DOS-based graphical libraries, which are no longer supported by the Windows-based compilers. The above limitations in the current software structure severely restrict the expandability of the software for handling a large freeway network with additional features such as traffic responsive ramp metering and HOV lanes.

640 KB Limitation

The current KRONOS, V8, had attempted to circumvent the 640-KB barrier by using overlays. Overlaying is a technique in which currently unused functions are swapped to disk to make room for functions in use. Using this scheme, the software larger than 640 KB can be run under DOS by keeping track of which functions are needed. Since disk access speed is substantially slower than that of RAM access, the software based on overlays show significant performance degrading as well as the difficulties in maintenance. In this research, the entire code of the simulation module in Kronos was converted to the 32-bit operations mode under the Windows environment. Further, a new Windows-based user interface was developed using the Boralnd Delphi Windows tool kit (Borland, 1996).

Static Memory Allocation

All the previous versions of the simulation module made exclusive use of static memory allocation. In static memory allocation, the size of all the data elements in the source code is predetermined, e.g., the length of all the on ramps, the maximum number of DXs on a freeway and the number of weaving sections, etc. By fixing the size of variables statically without considering the actual geometrics of the freeway section to be simulated, it is possible to waste the available resource, thereby unreasonably limiting the simulation capabilities. For example, if the programmer had pre-specified 30 on ramps and 30 off ramps, each 2000 ft. long, it would be impossible to simulate a case with 29 off ramps and 31 on ramps with the length of 500 ft. This is due to the fact this case exceeds the predefined maximum of 30 off ramps, even though 30 * 200 DX + 30 * 200 DX = 120,000 units of memory, which is much more than 29 * 50 DX + 31 * 50 DX = 30,000 units of memory. None of the "unneeded" 90,000 units of memory is available to the 31st off ramp, because it is predetermined that all off ramps are 2000 ft. long, and that only 30 off-ramps are allowed.

Using static allocation, in order to support one off ramp that is 2000 ft long, *all* off ramps must have enough data storage for 2000 ft. So, memory is wasted whenever a 500-ft off ramp is used, since all off ramps were defined to be 2000 ft. long. That memory is also not available to any other data element in the program. This inefficient use of memory, when combined with the 640-KB barrier, made it impossible to simulate large freeway networks with many additional features.

II. 2 Redesign of Data Structure with Dynamic Memory Allocation

The previous versions of KRONOS did not make use of records. Unlike arrays, which can hold only homogeneous data, records can hold heterogeneous data. This is very useful when different types of data are strongly related. For example, under previous versions, the following arrays existed.

```
float On_Ramp_Density[2][MAX_NUM_ON_RMPS][MAX_NUM_RMP_DXS + 1],
      On_Ramp_Speed[2][MAX_NUM_ON_RMPS][MAX_NUM_RMP_DXS + 1],
      MAX_NUM_ON_RMPS_Q_Size[MAX_NUM_ON_RMPS],
      On_Ramp_Capacity[MAX_NUM_ON_RMPS];
int   Num_Merging_Lanes[MAX_NUM_ON_RMPS],
      On_Ramp_Length[MAX_NUM_ON_RMPS];
```

These would be accessed as

```
On_Ramp_Density[0][Ramp_Num][0] = 15.64;
On_Ramp_Capacity[Ramp_Num] = 1500;
```

So, there is a separate array for each data item associated with an on-ramp and each data array has its own indexing and naming scheme. When there are large numbers of such arrays, it becomes very difficult to determine the relationships between the arrays. Records were devised to address this problem.

Since a single on ramp has a large number of data items associated with, it would be useful to make this relationship part of the data representation. Using a record, called a structure in C, this could be declared as

```
struct  On_Ramp_Struct_Type
{float  Density[2][MAX_NUM_RMP_DXS+1],
        Speed[2][MAX_NUM_RMP_DXS+1],
        Q_Size[2][MAX_NUM_RMP_DXS+1],
        Capacity;
  int   Num_Merging_Lanes,
        Length;
       } On_Ramp;
```

and could then be accessed as

```
On_Ramp.Capacity = 1500;
On_Ramp.Density[0][0] = 15.64;
```

In addition, it is then possible to declare an array of 30 On_Ramp structures as follows

```
struct On_Ramp_Struct_Type On_Ramp_Array[30];
```

A particular on-ramp with index Ramp_Num can be accessed as

```
On_Ramp[Ramp_Num].Capacity = 1500;
On_Ramp[Ramp_Num].Density[0][0] = 15.64;
```

The main advantage of using structures is that the relationship between data becomes apparent. Further, instead of dynamically allocating the entire on ramp arrays separately, the entire structure can be allocated at once, thus, significantly reducing the coding effort of converting from static to dynamic allocation. Also, the use of structures helps make indexing much clearer, since it is obvious which index is for the on ramp itself, and which are for data pertaining to the particular on ramp.

Dynamic Memory Allocation

Using dynamic allocation, the programmer doesn't specify the size of any data element that depends on the input data set. The program reads the input data set and allocates memory as needed. So, it would be possible to simulate a case with 120 off-ramps with the length of 500 ft or 30 with 2000 ft. While dynamic memory allocation makes it possible to use memory efficiently, it requires additional coding to read data set and allocate memory for each data item.

II.3 Other Enhancements in Software Structure for Performance Improvements

In addition to the elimination of the performance degradation due to overlays, the following significant performance improvements were made to the simulation module in this research:

- Elimination of post-processing of simulation results before entering the output module
- Faster writes of simulation data to disk
- Elimination of shifting data sets in memory

Efficient Post-processing of Simulation Results

Previously, the simulation module did not produce the output data sets in the format expected by the output module. As a result, at the end of each simulation, the output module had to convert the simulation output files into the file format it required. The simulation module now produces its results directly in the format expected by the output module, so no-post processing is necessary. On large KRONOS 8 data sets, this improvement alone can save 3-5 minutes of the output module execution time.

Faster Writes to Disk

When writing the simulation output data to the hard disk, the simulation module used to traverse the whole length of a given freeway each time it writes the output data, i.e., the speed, flow, and density data for each DX to the appropriate files. As a result, the procedure looked like this:

- On this freeway

- while not end of freeway
    - seek for the end of the instantaneous density file
    - write instantaneous density for this DX
    - seek for the end of the instantaneous speed file
    - write the instantaneous speed for this DX
    - .... and so on ...

Because disk searching is more time consuming than disk writing, it is much faster to traverse a given freeway N times, and write each file completely before going on to the next file, thus eliminating the disk searching all together. This new approach can be summarized as:

- On this freeway
    - while not end of freeway
        - write instantaneous density for this DX
    - while not end of freeway
        - write instantaneous speed for this DX
    - ... and so on ...

By processing an entire file once, the disk searching could be completely eliminated. This resulted in a 15% reduction in simulation time assuming a 15-minute aggregation interval for simulation output. The shorter the simulation output period, the greater the benefit.

Elimination of Copying Data Sets

The current numerical simulation methodology in Kronos uses the data in the previous time step to compute the data in the current time step. This resulted in all arrays having a final time index, i.e., DT, so that we can index density[dx][dt]. If we assume that the previous DT is dt = 0, and the current DT is dt = 1, then at the end of each DT, we have to copy all of the data from density[dx][1] to density[dx][0], so that density[dx][current] is available for the next DT's simulation. For a large freeway network, it would require a lot of memory copies, thus substantially increasing the execution time considering the fact that currently DT is one second.

However, if we don't assign a fixed numeric value to previous and current, we can simply swap their values, accomplishing the same results.

Old scheme:

```
DT = t
    compute q[all_dx][1], k[all_dx][1], u[all_dx][1]
    copy q[all_dx][1] to q[all_dx][0]
    copy k[all_dx][1] to k[all_dx][0]
    copy u[all_dx][1] to u[all_dx][0]
DT = t+1
```

New scheme:

```
DT = t
    if previous_dx = 0
        previous_dt = 1
        current_dt = 0
    else
        previous_dt = 0
        current_dt = 1
compute q[all_dx][current_dt], k[all_dx][current_dt], u[all_dx][current_dt]
DT = t+1
```

By simply swapping the index every DT, we can avoid the time consuming copy operation by simply redefining which DT index is current and which DT index is previous.

Reduction of the Size of Output Files

Previous version of KRONOS included extra spaces in the simulation output files, where the output data is written one item per line. Since the C function “fscanf”, which performs the file read operations in KRONOS, treats any positive integral number of spaces as one space and uses CR/LF as the data separator, there is no need to pad the file with extra spaces.

The extra spaces arose from improper output formatting. A floating-point value can be printed by specifying no formatting, i.e.,

```
printf("Here is your floating point variable %f", Density)
```

or by explicitly specifying the field width and the number of decimal places by width.decimal_places such as

```
printf("Here is your floating point variable %6.2f", Density)
```

which prints the floating point value in a field with a width of at least six, and with exactly two decimal places.

All of the output files had excessively large field widths, which resulted in padding the field with leading spaces. To avoid this problem, only the desired number of decimal places was specified. This produces a flexible field width with the desired accuracy, with no leading spaces, as shown below.

```
printf("Here is your floating point variable %.1f", Density)
```

For large input cases, this resulted in a 10-25% reduction in file size, depending on whether the file was a speed, density, or flow file.

Replacing Redundant Array Indexing With Local Variables

Another performance improvement came from replacing repeated array indexing with local variables. The numerical method adopted in Kronos uses a central difference scheme across 3 adjacent dxs. This bred the notation Prev_DX, Curr_DX, and Next_DX in order from upstream to downstream. Since the density and speed values corresponding to these locations are accessed repeatedly, performance gains could be realized by replacing array indexing such as

```
for ( Curr_DX = 0;
    Curr_DX < Last_DX;
    Curr_DX++)
{
  if ( RHS_Density[0][Current_DX] <= qkcurves[zno][0].kcr )
   qofframp = Diversion_Rate[dc] * ( RHS_Density[0][Current_DX] *
   RHS_Speed[0][Current_DX] - Exit_Demand[dc] ) / 100 + Exit_Demand[dc];
  else
   qofframp = Diversion_Rate[dc] * ( qkcurves[zno][0].qmax -
        Exit_Demand[dc] ) / 100 + Exit_Demand[dc];
  if ( RHS_Density[0][Current_DX] <= qkcurves[zno][0].kcr )
  {
   q_avail = RHS_Density[0][Current_DX] * RHS_Speed[0][Current_DX];
   q_allow1 = qkcurves[zno][0].qmax;
  }
  else
  {
   q_avail = qkcurves[zno][0].qmax;
```

```
    q_allow1 = RHS_Density[0][Current_DX] * RHS_Speed[0][Current_DX];
  }
 }          /* end of for all dx */
```

with a scheme such as

```
  Mainline_Kcr = qkcurves[zno][0].kcr;
  Mainline_Capacity = qkcurves[zno][0].qmax

 for ( Curr_DX = 0;
     Curr_DX < Last_DX;
     Curr_DX++)
 {
  K_Curr = RHS_Density[0][Current_DX];
  U_Curr = RHS_Speed[0][Current_DX];

   if ( K_Curr <= Mainline_Kcr )
    qofframp =   Diversion_Rate[dc] * ( K_Curr * U_Curr - Exit_Demand[dc] ) /
              100 + Exit_Demand[dc];
   else
    qofframp =   Diversion_Rate[dc] * ( Mainline_Capacity -
              Exit_Demand[dc] ) / 100 + Exit_Demand[dc];

   if ( K_Curr <= Mainline_Kcr )
   {
    q_avail = K_Curr * U_Curr;
    q_allow1 = Mainline_Capacity;
   }
   else
   {
    q_avail = Mainline_Capacity;
    q_allow1 = K_Curr * U_Curr;
   }
  }     /* end of for all dx */
```

This is a small excerpt of a typical section of code. As can be seen, there are a number of array variables that are repeatedly accessed in each iteration. Since the program must determine the index values at run time, and compute the address using base address + index * size of element, there is run time overhead associated with array indexing. By copying the array value into a local, unsubscripted variable, the overhead is required only once at assignment to the local variable, and the local variable can then be repeatedly accessed without the overhead of indexing.

An added benefit is more readable code. More meaningful local variable names can be used which describe the process more thoroughly. This greatly improves the readability of the code, and helps separate the algorithmic description from the data structure implementation. This is clearly useful should the algorithm or the data structure need to be changed independent of the other.

The data structure is not woven into every line of the program, and the algorithm is then less dependent on the structure. This greatly increases the ease of modification.

III. ENHANCEMENT OF SEGMENT-BASED MODELING STRUCTURE

III.1 Issues in the Current Modeling Structure

The modeling structure of Kronos is based on the geometric types of freeway segments and different simulation procedures have been developed for each segment type using a macroscopic modeling approach. In the current version of Kronos, a freeway is divided into 24 different types of segments and each segment is further discretized into 100-ft increments, called DXs. Figure 2-1 shows the current definition of geometric segment types used in Kronos 8. As illustrated in this figure, most segments have leading and/or lagging pipeline segments where traffic flow either splits from single to two flows or merges from two to one flow. This is to reflect the different flow behavior at the right or left most lanes associated with the entrance/exit ramps. For example, drivers usually complete their lane-changing maneuver before they reach an off-ramp, the flow split in the mainline near an off-ramp can be assumed to happen at the pipeline portion before the diverging point. Further, the drivers entering mainline from an entrance ramp try to change the lanes after they passed the merging point between the on-ramp and mainline. Therefore, the pipeline portion of each segment type has an internal boundary where a single flow is splited into two flows or two flows merge into one flow.

While these internal boundaries require complicated numerical treatments, in the current version of Kronos, those internal boundaries have been handled within each segment type, which resulted in the duplication of the same boundary treatment code at each segment simulation module. This redundancy has created unnecessary complexity in maintaining and upgrading the simulation module. Further, to facilitate the connectivity between two adjacent segments, the current version requires each segment to start and end with the single flow DX, which has imposed certain minimum length limitations for each segment type. For example, the simple on-ramp and a simple weave segments need 2 Dxs both at the starting and ending pipeline portions. Currently the user interface internally converts a given freeway into the required segment types, which has been a time-consuming, complex process and often caused run-time error. Also, this process is not completely transparent to user, as the resulting segment types could be different from the initial segment definitions the user used to build a freeway on the screen. Further, the use of single capacity value for each segment sometimes does not allow the user to apply different capacity values for the short pipeline near a weaving or diverging section.

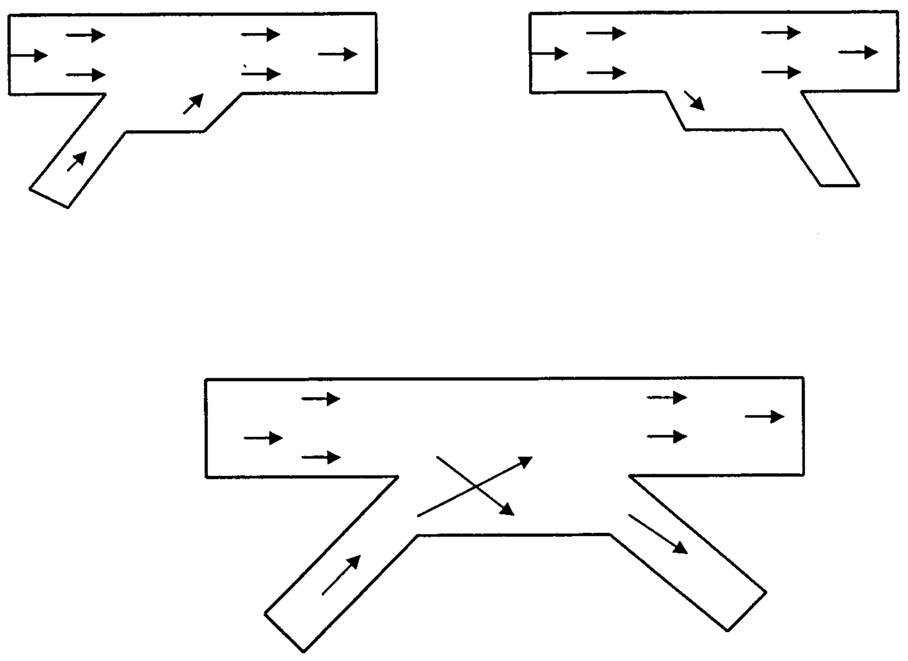


Figure 3-1. Segment types with lead/lag pipelines in Kronos 8

III. 2 Development of new pipeline-segment types

For more efficient treatment of internal boundaries and simplification of the source code maintenance, a new set of pipeline segment types with various internal flow split configurations was developed in this research. Figure 3-2 shows the types of pipeline segments developed for different internal flow split/merge combinations, i.e.,

ONE-TO-ONE-PIPE	-	One flow in and one flow out
ONE-TO-TWO-PIPE	-	One flow in and two flows out
TWO-TO-ONE-PIPE	-	Two flows in and one flow out
TWO-TO-TWO-PIPE	-	Two flows in and two flows out.

Based on the type of the adjacent segments, an appropriate type of pipeline can be determined by the simulation module. The use of the new pipeline segment types significantly reduced the redundancy in the source code by treating the flow split/merge process within the pipeline segments. Further, it enables the simulation module use much shorter minimum lengths for freeway segments than those of the current version.

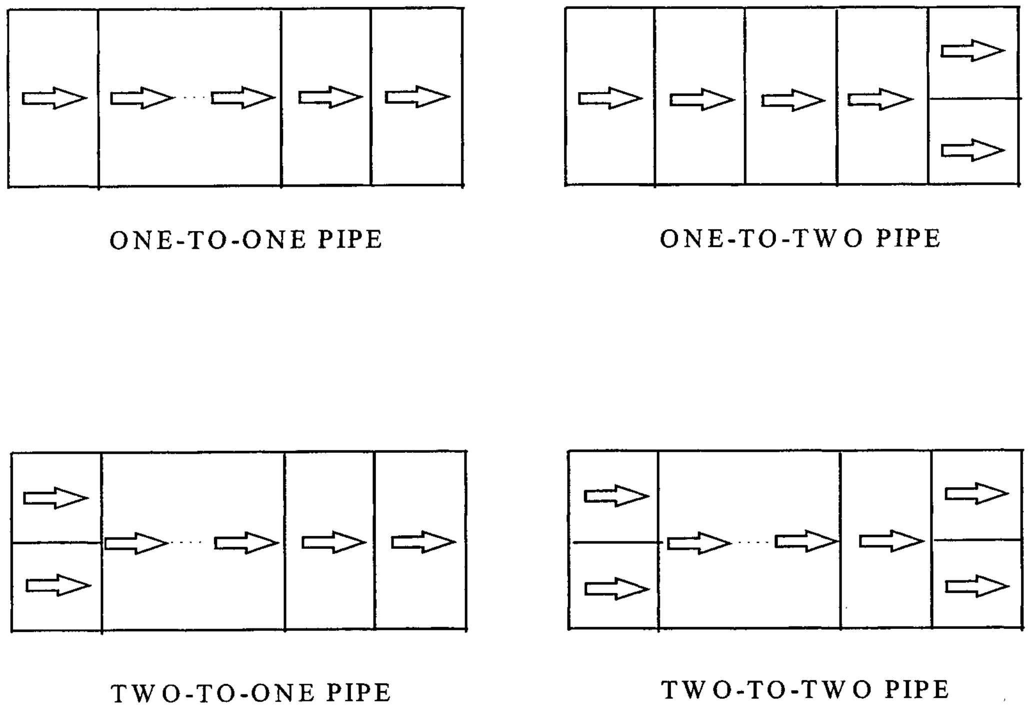


Figure 3-2 New pipeline segment types

III.3 Simulation procedures for new pipeline segments

This section describes the flow simulation procedure for each pipeline segment shown in Figure 3-2. The following notations are used in describing the simulation procedure fore each segment.

q^t_{j-1} = Flow in j-1th DX at t

q^t_{j+1} = Flow in j+1th DX at t

k^t_{j-1} = Density of j-1th DX at t

k^t_{j+1} = Density of j+1th DX at t

q_{main} = Flow in mainline region

q_{adj} = Flow in adjacent region

a. One-One-One:

This is same as the simple PIPELINE segment in Kronos 8. The numerical scheme used for this segment is as follows;

$$k^{t+1}_j = \frac{1}{2}(k^t_{j-1} + k^t_{j+1}) + \frac{DT}{2DX}(q^t_{j-1} - q^t_{j+1}) \qquad \text{Eqn 1}$$

b. One-One-Two:

In this segment, the flow in j+1st DX is computed as the sum of the flows in its adjacent and mainline regions.

$q^t_{j+1} = q_{main} + q_{adj}$

$q^t_{j+1} = \min(q_{\max}, q^t_{j+1})$

$q_{\max}$ = Capacity of j+1st DX

Using the q-k curve of j+1st DX, find equivalent density(k^t_{j+1}) corresponding to this total flow. Density in the mainline region is compared with critical density of the mainline q-k curve to decide whether q^t_{j+1} belongs to congested or uncongested region of the q-k curve. After obtaining the equivalent density of j+1st DX, we can apply simple Lax equation as in Eqn. 1 to determine the density in the current DX.

c. One-Two-Two:

This DX combination occurs for a flow-split DX, i.e., DX in which a single flow in previous DX is split into two. We use the same method as described above to determine the equivalent density(k_{j+1}^{t}) in the next DX. Using this density and the density in the previous segment, we can determine the density in the current DX using Lax equation (Eqn 1). This density has to be distributed between the mainline and adjacent portions, based on the quantity of flow in the two regions.

Flow in next DX is $q_{j+1}^{t+1} = k_{j+1}^{t+1} * u_{j+1}^{t+1}$

Flow-split

Flow in adjacent and mainline portions is determined by the type of segment downstream of current segment. If downstream segment is an on ramp, we distribute the flow equally among the mainline and adjacent lanes. Number of lanes in the adjacent region of current segment is same as the number of adjacent lanes in the next segment. For example, if next segment is a simple on/off ramp or a simple weave, then number of adjacent lanes is one.

$$q_{adj} = \frac{q_{j+1}^{t+1}}{Number\ of\ lanes} x\ Number\ of\ Lanes\ in\ Adjacent\ \mathrm{Re}gion$$

$$q_{main} = q_{j+1}^{t+1} - q_{adj}$$

If next segment is an off ramp, then exit demand onto off ramp has to be taken into consideration while calculating q_{adj}. The following sequence of operations is performed to determine the flow in adjacent and mainline regions.

```
$if\ (k_{j-1}^{t} < k_{cr,j-1})$
$q_{avail} = k_{j-1}^{t} * u_{j-1}^{t}$
$else$
$q_{avail} = q_{\max,j-1}$
$Exit\_Q = min(exit\ demand,\ q_{avail})$
```

$$q_{adj} = \frac{q_{j+1}^{t+1}}{Number\ of\ lanes} x\ Number\ of\ Lanes\ in\ Adjacent\ \mathrm{Re}gion$$

$$q_{adsj} = \max(q_{adj}, Exit\_Q)$$

$$q_{main} = q_{j+1}^{t+1} - q_{adj}$$

If downstream segment is an exclusive off ramp, only exit demand occupies the adjacent portion.

$$q_{adj} = Exit\_Q$$

$$q_{main} = q_{j+1}^{t+1} - q_{adj}$$

Using q-k curves of the two regions, we can determine the densities in these regions for the two flows computed above.

d. Two-One-One:

This combination can occur in the second DX of a TWO-TO-ONE or a TWO-TO-TWO PIPE segment. First, compute the total flow in j-1st DX as follows:

$$q_{j-1}^{t} = q_{main} + q_{adj}$$

$$q_{j-1}^{t} = \min(q_{\max}, q_{j-1}^{t})$$

q_{max} = Capacity of j-1st DX

Using q-k curve of j-1st DX, determine the density corresponding to this flow. Density of mainline region is compared with mainline critical density to decide whether q_{j-1}^{t} falls in congested or uncongested region of the mainline q-k curve. After obtaining the equivalent density of j-1st DX, we can apply simple Lax equation(Eqn. 1) to determine the density of the current DX.

e. Two-Two-One:

This combination may occur in the first DX of a TWO_TO_TWO_PIPE or TWO_TO_ONE_PIPE. Using the same procedure as explained in the previous section, compute the equivalent density in the previous DX. Apply Eqn 1 to compute the equivalent density k_{j-1}^{t+1} in the current DX. From this density, find the total flow in the current DX as $k_{j-1}^{t+1} * u_{j-1}^{t+1}$.

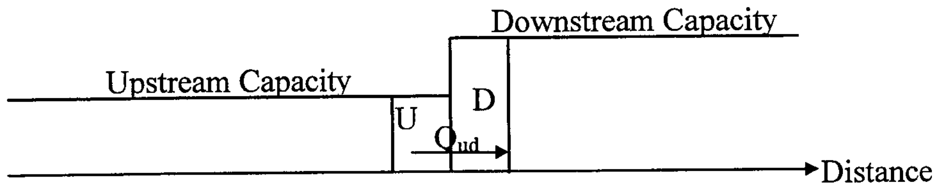
Distribute this total flow among adjacent and mainline regions based on the number of lanes in each region. Using the q-k curves of the two regions, we can determine the densities in the two regions corresponding to the two flow values computed above.

f. Two-One-Two:

With the current KRONOS model, this combination can occur only in a TWO-TO-TWO-PIPE segment. This is only possible if length of the current segment is 3DX. To calculate density of the current DX, the equivalent densities of the previous and next DXs have to be computed using the procedure explained in previous cases. Apply the simple Lax equation(Eqn 1) to obtain density of the current DX.

III.4 Boundary treatments between two segments with different capacities

The basic numerical scheme adopted in Kronos to simulate traffic flows assumes constant external conditions for flows, i.e., same flow-density relationships. When two segments have different flow-density relationships, the method caused some numerical error in terms of flow conservation at the boundary between two segments. To cope with this boundary error problem, the early version of Kronos 8 used a special scheme called "available-allowable", where the flow value crossing the boundary between two segments is determined as the minimum of the "available flow at the last dx of upstream segment " and the "allowable flow by the first dx of downstream segment" as follows:



$Q_{ud} = \min(Q_{AVAILABLE}, Q_{ALLOWABLE})$

if $(k_D > k_{D\_CR})$

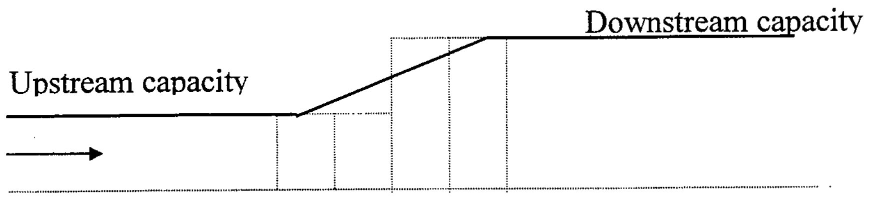
$Q_{ALLOWABLE} = k_D * u_D$

```
    else
        $Q_{allowable} = Q_{cap\_d}$ ; capacity of downstream segment
and
    if ($k_U > k_{U\_CR}$)
        $Q_{available} = Q_{cap\_u}$ ; capacity of upstream segment
    else
        $Q_{AVAILABLE} = k_U * u_U$
    where, k, u denote density and speed respectively.
```

While the above method worked well in conserving the amount of flows crossing the internal boundary between two segments, it did not properly address the propagation of the shock wave from the downstream segment. This resulted in the time delay for congestion to spill back toward upstream segment, which significantly affected the accuracy of the simulation results. In this research, a new method is developed to address both flow conservation and shock wave propagation.

Capacity Gradient Method

The new method developed in this research considers continuous capacity variation at the internal boundaries between two segments with different capacities, while the previous method assumes discontinuous capacity changes. This method is a compromise between the original numerical scheme and the previous method by assuring proper propagation of shock wave while minimizing the numerical flow conservation error at the internal boundaries because of the different capacities. The new method determines the capacity values of the boundary dxs by linearly interpolating the capacities of two segments as shown in the figure below. Using the modified capacity values, the normal numerical method is applied to the boundary dxs to estimate the traffic parameters.



The new method, called "Capacity Gradient Method", requires additional flow-density curves for each boundary DX to be stored. When this method was initially implemented in KRONOS 8 without using the DOS extender or dynamic memory allocation, a single q-k curve was recomputed for each boundary dx every dt. This caused excessive performance overhead, which resulted in additional modifications for the efficient treatment of the flow-density relationship data as described in the later chapter in this report.

IV. DEVELOPMENT OF FREEWAY NETWORK AND EXCLUSIVE HOV LANE SIMULATION PROCEDURE

IV.1 Overview of HOV lane and network simulation

High Occupancy Lanes (HOV) can be classified into two groups depending on the way they are separated from the normal lanes, i.e., exclusive HOV lane with physical barrier and diamond lane that does not have any physical separator. The exclusive HOV lanes can be treated as a separate freeway connected to the main freeway through on/off-ramps. In this research, a new simulation module was developed to treat a bi-directional freeway network including exclusive HOV lanes. The resulting network module can also simulate a ring road, which was not possible with the previous versions. Further, a new simulation procedure to handle the diamond HOV lanes without any physical barrier was also developed and incorporated into the simulation module. Due to the lack of the real data regarding the traffic demand for HOV lanes, qualitative testing was performed with hypothetical data.

IV.2 Development of new freeway data structure for network simulation

Freeway data structure in the current version

The current version of Kronos can handle three freeways, i.e., a main freeway and two merging/diverging freeways to/from the main freeway. However, the merging/diverging freeways were squeezed onto the end of the main freeway's data structure and a single-array was used by all three freeways. The mainline, merging, and diverging freeways were concatenated into a single static array as follows:

```
+------------------+---------------+----------------+
|      Main FW     |  Merging FW   |  Diverging FW  |
+------------------+---------------+----------------+
```

If this implementation had used a separate array for each of the freeway types, the resulting structure would have been:

```
+------------------+
|     Main FW      |
+------------------+
+------------------+
|    Merging FW    |
+------------------+
+------------------+
|   Diverging FW   |
+------------------+
```

However, doing this would have greatly reduced the maximum length of the main freeway. This was all done using static allocation, so three arrays, each 1/3 of the length would have been declared. As a result, even without any merging or diverging freeway to simulate, the "unneeded" memory they consumed would be unavailable to the main freeway. This would have meant that even in the absence of merging and diverging freeways, the main freeway could have only been about 6 miles long, instead of 18 under the "all in one array" approach. So, the "all in one" array approach met the goal of efficient memory use, essentially doing a form of dynamic allocation between the main, merging, and diverging freeways within the limits of the statically allocated array.

However, the existing data structure is too inflexible to support generic freeway network topologies. Using this scheme, the per DX flow, speed, and density arrays *assumed the connection structure*. The order of data in the array was based on the assumption that only a main, merging, and diverging freeway would exist. In an arbitrary freeway network, there is no notion of a main freeway. Also, in an arbitrary network, there may be freeways, which intersect other freeways multiple times, or even beltways, which intersect themselves. There is no way to map such a topology into the linear freeway by freeway array, other than arbitrarily allocating the multiple freeways as shown in the following figure.

```
+------+------+------+------+------+------+-----+
| FW 0 | FW 1 | FW 2 | FW 3 | FW 4 | FW 5 | FW 6|
+------+------+------+------+------+------+-----+
```

However, if the freeways are ordered arbitrarily like above, we have lost the assumed connection structure imposed by the original implementation, so we have no way of knowing how freeway 3 connects to freeway 6, if at all. To be able to handle a network of multiple freeways, it is crucial to develop an efficient and generic data structure that can represent the relationship between two connected freeways more realistically.

Generic Freeway Representation

The new data structure developed in this research separates the connection information from the freeway representation. There are now distinct data structures for each freeway, rather than one shared by all. In addition, no connection structure is forced on the freeway data organization. Removing the connection information from the freeway allowed for the idea of a *generic freeway*. This generic representation attempts to capture only the geometric description of the freeway, not the geometric and interconnection information as was previously implemented. The main benefit of this is that a generic freeway structure and structure allocator could be designed without regard for how the freeways were interconnected. Each freeway is examined in light of its boundaries. A boundary is defined as the point where vehicles enter or exit a freeway. The list of all possible boundaries includes:

- First DX of the first segment of freeway (entering volume)
- Last DX of the last segment of freeway (exiting volume)
- First DX of all on ramps (entering volume)
- Last DX of all off ramps (exiting volume)

A connection can exist between any pair of boundaries, as long as the flow direction across the boundary is the same for both sides of the boundary. This means that an exiting flow boundary has to be connected to an entering flow boundary, and vice versa. Connecting two exiting flow boundaries (two off ramps for example) would produce a connection, which has no place in a real freeway network. Some example connections are:

- Beltway: Last segment of freeway to first segment of freeway
- Merging Freeway: Last segment of freeway to freeway on ramp
- Diverging Freeway: Off ramp to first segment of freeway
- HOV Freeway: Off ramp to first segment on the upstream end and last segment to on ramp on the downstream end
- Subsection to Subsection: A single freeway could be broken into logical subsections, with connections between the last DX of one section and the first DX of the next.

Any connection which meets the constraints of legal traffic direction (exiting to entering or entering to exiting) and the same number of lanes are allowed. It is also possible to simulate a network of unconnected freeways. This would be useful for batch simulation of a large number of freeways overnight, for example. In addition, the support for unconnected freeways also allows for the simulation of two freeway networks.

Data structure for connection points between two freeways

Four kinds of connections are possible between two freeways:

Off Ramp To On Ramp (freeway interconnection)

Mainline To On Ramp (freeway termination/merging freeway)

Off Ramp To Mainline (freeway origination/diverging freeway)

Mainline To Mainline (beltway/ring road)

A connection can occur at any freeway boundary. A boundary is defined as a freeway element, which has one side, at either flow entry or flow exit, open. This includes the upstream side of the first mainline segment, the downstream side of the last mainline segment, the downstream side of any off ramp, and the upstream side of any off ramp.

In the new data structure, a connection is defined by the participating sections as follows:

```
struct Connection {
 int  Upstream_FW_Num,
      Upstream_Sec_Num,
      Upstream_Sec_Type.
      Downstream_FW_Num,
      Downstream_Sec_Num,
      Downstream_Sec_Type,
      Type;
};
```

The Sec notation is a short hand for Section. Since the value could apply to a segment, an on ramp, or an off ramp, and each of these is numbered independently, the connection Type is stored to indicate which data structures are relevant for this connection. Section types are stored to help ensure that the geometrics to be simulated are the same as the geometrics for which the connection was connected. This helps to prevent the simulation of erroneous scenarios.

The major improvements provided by the above structure are the two new connection types. These allow for simulation of ramp to ramp connections, which have typical cloverleaf type interchanges, and mainline to mainline connections, which allow for simulation of beltways, which exist in many metropolitan areas. The 'mainline to mainline' connection type can also be used to divide a single freeway into logical subsections that are then connected end to end. In addition, A network can consist of both connected and unconnected freeways. This allows for simulation of groups of networks, which is useful for batch simulation.

With the new freeway and connection data structure, it's now much simpler to check if a DX is the last DX of a freeway. For example, the logic to check for the last dx of a merging freeway type looks like this

```
if(   ( Current_DX == FW->Last_DX ) &&
      ( Seg->CONNECTED )
   )
```

This new scheme only has to check if this is the last dx and if this segment is connected. Since only the terminal segments can be connected to other freeways, it is only necessary to check if this is the last dx of the freeway, and if it is connected to another freeway. This logic is clearer and more concise.

The new freeway data structure has the following limitations in terms of the number of freeways and connection points:

30 freeways

500 connections

The first limitation arises from the maximum number of file handles provided by the installed Borland libraries. This maximum number of file handles limits the number of files that can be open at once. This is due to the limitations in the current libraries using static allocation for file handles. While this restriction can be relaxed if necessary, most practical cases would not include more than 30 freeways. The second limitation arises from the fact that the input module, unlike the simulation module, still uses static allocation, so some absolute maximum value must be defined in the input module. This can also be altered by recompiling the input module with a more suitable value. Again, it is unlikely that this value will prove to be a limitation in the short term, but it will be easy to adjust it to any required value within the confines of memory.

IV. 3 Input/Output file structure for network simulation

Under the new scheme, each freeway is treated as a main freeway in the old KRONOS 8 sense. Each freeway has its own set of geometry and demand files (as well as the files necessary for ramp metering discussed elsewhere). There is also a single connection file that describes the freeways involved in the simulation case, as well as the connections between them. Since geometrics rarely change, but a large variety of demand and metering scenarios could be run against the same geometrics, the new file layout uses separate files for each kind of data for each network. This is a departure from all previous version of KRONOS which had all of the data in a single file. The input files for a network include the following files:

*.NET	Network file specifying freeways in this network, and their interconnection
*.GEO	Geometry for each freeway in the network
*.DEM	Demand for each freeway
*.FIX	Fix rate metering rates for freeway
*.MET	Automatic Rate Selection Metering Data for each network
*.STA	Station data for this freeway
*.INC	Incident file per freeway
*.DET	Detector data for this freeway
*.ZON	Metering zones for this network case

Using this approach, the metering policy, whether fixed rate or Automatic rate selection, can be modified without changing the geometrics. Likewise, many sets of demand data can be run against the same geometry without modifying the geometry data file. When tools exist to convert detector and station data files to demand files, this will be much easier when only the .DEM file must be created. Under the previous scheme, the main, merging, or diverging freeway file would have to have been parsed to find the demand data, and then modified to reflect the new demand.

IV. 4 Network simulation process

Because the freeway data files for a given case could be copied between subdirectories, the path to the input, output, and case file are passed from the input to simulation module via the environment variables, rather than hard coded in the network case file. The simulation module gets the information about the paths to the files from the environment variables before it reads the network file. The network file contains the data regarding the number of freeways in the network to be simulated, the filename prefix for each of the freeways for the given network, and the number of segments of each type on each freeway in the network. Finally, the network file contains the connection table that identifies the physical connection points between the boundaries of two freeways.

The simulation module opens the network file, reads the number of freeways in the network, and immediately allocates that many empty instances of the freeway structure. Once this is done, the number of segments of each type is read for each freeway, and memory for each segment's parameters, except data arrays, is allocated, along with associated on and off ramps. Finally, the connection table is read, and the connections are mapped into the segment data structures.

Once the overhead of allocating network and segment structures has been done, the freeway geometrics are read. Once the geometrics have been read, data arrays of the appropriate length are allocated for all segment and their associated on and off ramps based on the geometric information for this case.

After geometrics, connections, and data arrays are set up for all given freeways, we are ready to simulate the case. We loop through all the freeways in a network, traversing every dx in each freeway before proceeding to the next. Each freeway can have distinct output options, as well as simulation result periods. One freeway could have instantaneous and average flow output every 5 minutes. Another in the same network could have MOE's and only average flow calculated every 15 minutes.

Simulation at Freeway Connection Points

The simulation at the connection points between two freeways makes use of the capacity gradient, if there is a capacity difference between two connected segments, and the regular LAX method. Each segment or ramp connected, hereafter referred to as a section, has a pointer to the appropriate "connected DX" set at the time the connection table is read. This connected DX pointer is set for the speed, flow, and density variables during the initialization period to save the overhead of re-computing it for every DT, i.e., one second. Once all the data relationships between ramps and segments connected are set up, the simulation at the connection points can be performed using the simulation procedure for the simple pipeline case.

IV. 5 Testing network simulation module

In this section the network simulation module was tested with a hypothetical ring road geometry. More rigorous testing with real traffic data will be performed in the subsequent phase of this research. The beltway consists of three pipeline segments with the total length of 20,000 feet and the first pipeline segment is connected with the last pipeline segment. A platoon of vehicles was concentrated near the middle of the freeway at t = 0. Figure 4-1 shows the simulation results indicating density variation through time. The flow rates are seen to slowly distribute themselves uniformly over the freeway in time as they continuously loop around the beltway. Also, the effects of platoon reached to the both boundaries, i.e., the farthest points from the initial platoon location, at t= 10 minute.

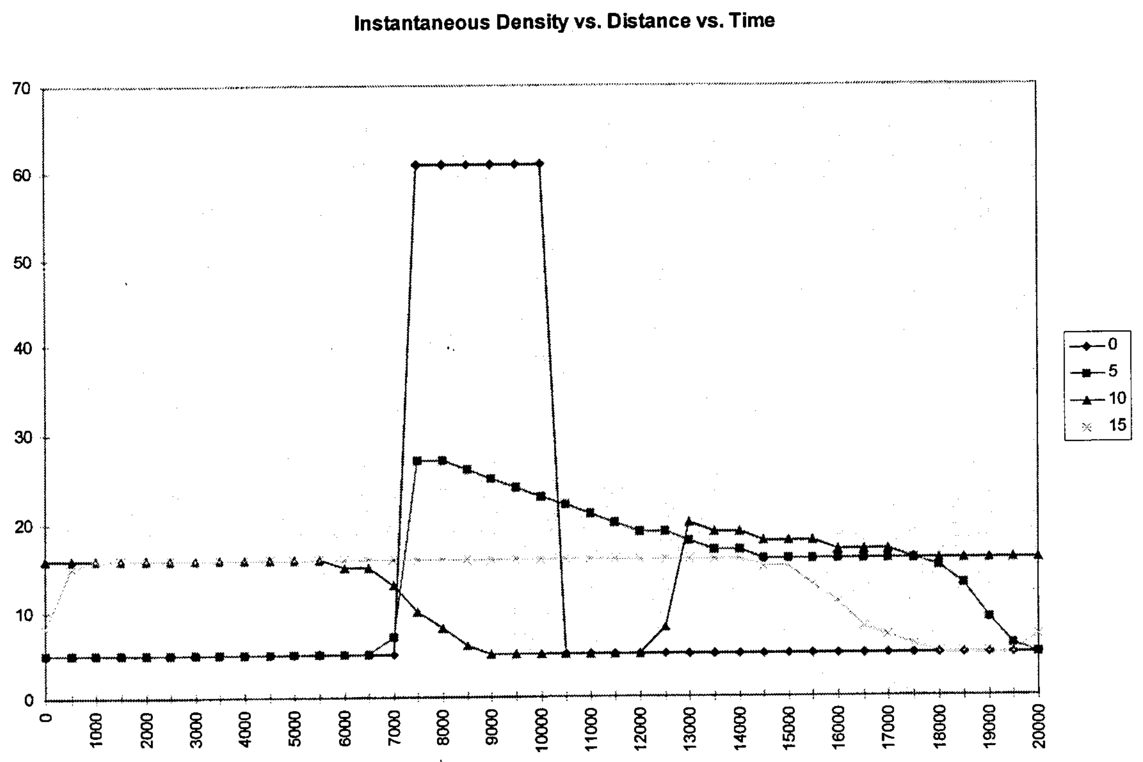


Figure 4-1 Test results for the network module with a hypothetical example of a ring road

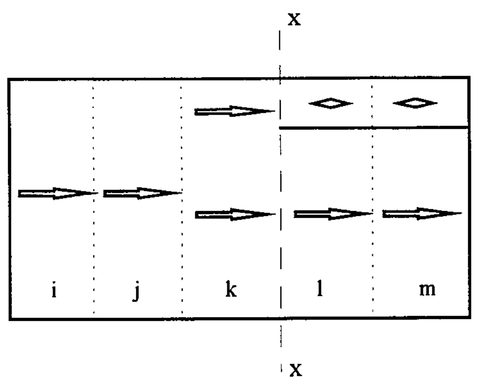
V. MODELING NON-EXCLUSIVE HOV LANE TRAFFIC FLOWS

V.1 Overview of Diamond HOV lane flow simulation

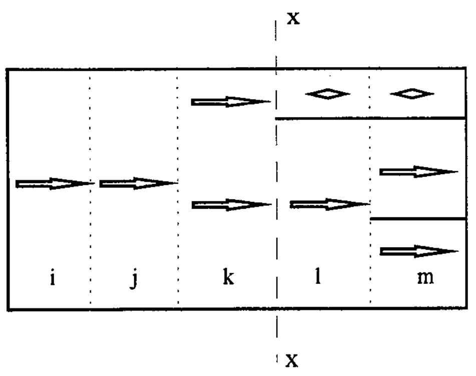
The diamond HOV lanes do not have physical barrier that separates them from the normal lanes. While drivers are allowed to enter the diamond lane or divert from it at any location, based on the observation of drivers' behavior and considering the limitations in the macroscopic modeling approach adopted in this research, it is assumed that there are certain merging/diverging points between the diamond and normal lanes depending on the geometric conditions, i.e., the location of entrance and exit ramps. This assumption is based on the general behavioral characteristics of drivers, i.e., HOV lane users tend to enter HOV lanes as quickly as possible. Therefore, the diamond HOV lane is treated as a freeway section with multiple merging/diverging points interacting with the normal lanes and a set of new simulation procedures was developed for the beginning/ending and merging/diverging segments between the diamond and normal lanes. Figures 5-1 and 5-2 illustrate the geometric types for the beginning and ending segments of the diamond lane section. At the beginning segment, the diamond lane entry demand needs to be specified by user. Further, for each on and off ramp, the HOV lane demand component also needs to be provided for the simulation module.

V. 2 Modeling beginning and ending segments for Diamond lanes

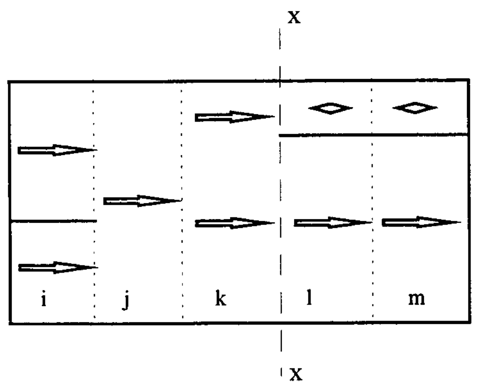
It is assumed that a HOV lane can only start within a pipeline section or with a lane-add segment. Similarly, the HOV lane can only end within a pipeline or with a lane drop section. It is also assumed that the HOV is on the left-hand side of the freeway. As described chapter III, simulation module internally represents a pipeline segment as one of the four pipeline sections based on its preceding and succeeding segment types. Therefore, while devising the flow model for the starting or ending segments, we have to consider all these four pipeline segments and the lane add/drop sections. Figures 1 & 2 respectively show the valid starting and ending segments of a HOV region. The following length restrictions have been imposed for each of these segments.



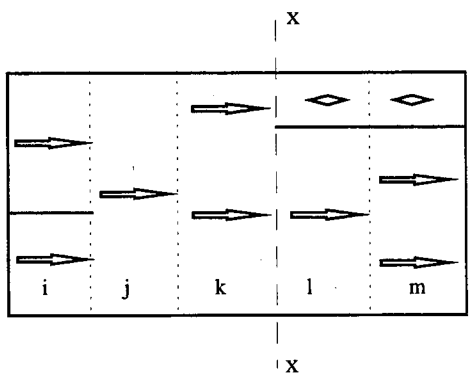
a. ONE-TO-ONE-PIPE



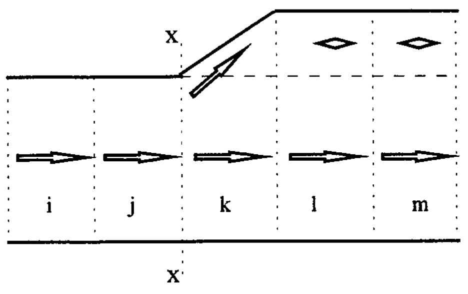
b. ONE-TO-TWO-PIPE



c. TWO-TO-ONE-PIPE

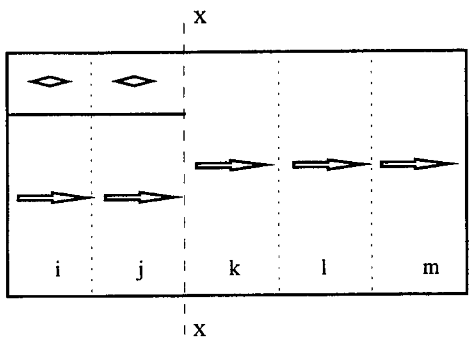


d. TWO-TO-TWO-PIPE

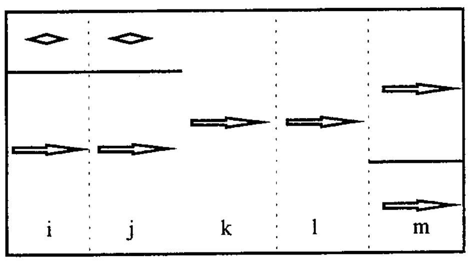


e. LHS_LANE_ADD

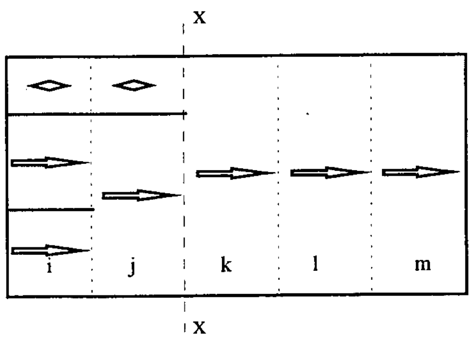
Figure 5-1 Diamond HOV lane starting segment types



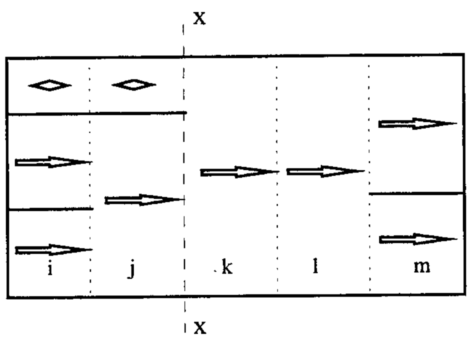
a. ONE-TO-ONE-PIPE



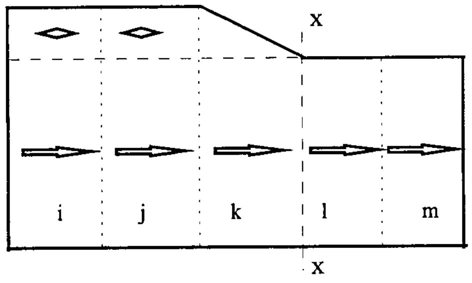
b. ONE-TO-TWO-PIPE



c. TWO-TO-ONE-PIPE



d. TWO-TO-TWO-PIPE



e. LHS_LANE_DROP

Figure 5-2 Diamond HOV lane ending segment types

The starting and ending pipeline segments should be of at least 5DX length.

- For HOV starting segment, the HOV region should start at least 2DXs before the end of the pipeline
- For HOV ending segments, the HOV region should end at least 2DXs after the beginning of the segment.

Modeling of flow in each of these segments is described in the following. The section x-x in the figures indicates the boundary between HOV and normal lanes.

ONE-TO-ONE-PIPE:

Figure 1(a) shows the flow pattern in this segment type. The total mainline flow is split into HOV flow and mainline flow at section x-x. The amount of flow that should be let into HOV lanes is given as an input to the program. The following procedure describes the method used to calculate the density of kth DX.

1. Find the available flow, $Q_{\text{available}}$ in kth DX as

   $if\ (k_k < k_{cr,k})\ Q_{avail,k} = k_k * u_k$

   else

   $Q_{avail,k} = Q_{\max,k}$

2. Let Q_{hov} be the HOV entry demand at the upstream as given by user. Let Q_{hov_out} is the flow that should enter the HOV lane.

   $Q_{hov\_out} = \min(Q_{avail,k}, Q_{hov})$

3. Let $Q_{allow,hov}$ is the allowable flow into 1st DX of HOV lane. It is computed as follows:

   $if\ (k_{1,hov} < k_{cr,1,hov})\ Q_{allow,hov} = Q_{\max,1,hov}$

   else

   $Q_{allow,hov} = k_{1,hov} * u_{1,hov}$

   $Q_{hov\_out} = \min(Q_{hov\_out}, Q_{allow,hov})$

where

$k_{cr,k}$ = Critical density of kth DX

$Q_{max,k}$ = Capacity of kth DX

$k_{l,hov}$ = Density of kth DX in HOV region

$k_{cr,l,hov}$ = critical density of lth DX in HOV region

$Q_{max,l,hov}$ = Capacity of lth DX in HOV region

4. The remaining of the available flow in kth DX should go to mainline region. The remaining flow is

$$Q_{remain} = Q_{avail,k} - Q_{hov\_out}$$

We have to compute the allowable flow, $Q_{allow,l,ml}$ into mainline region. Actual flow that is entering the mainline region is computed as

$$Q_{ml\_out} = min(Q_{remain}, Q_{allow,l,ml})$$

5. Density in kth DX for the next DT is now computed using modified Lax algorithm

$$k_k^{t+1} = \frac{1}{2}(k_j + k_k) + \frac{DT}{DX}(\frac{1}{2}(Q_j + Q_k) - (Q_{hov\_out} + Q_{ml\_out})$$

Density Computation for lth DX in HOV region:

$$k_{j,hov}^{n+1} = \frac{1}{2}(k_{l,hov}^t + k_{m,hov}^t) + \frac{DT}{DX}(Q_{hov\_out} - \frac{1}{2}(Q_{l,hov} + Q_{m,hov}))$$

Similar equation can be used for calculating mainline density of lth DX.

For all other HOV DXs in this segment, the simple Lax algorithm can be used. All other DXs in the mainline region fall into one of the DX combinations described in chapter III.

V.3 Merging/diverging between HOV and normal lanes

Each on-ramp area in a HOV region has a predetermined DX which is designated as its 'HOV merge point', where the HOV demand flow from this ramp enters the HOV lane. Similarly, each off ramp segment in a HOV region has a predetermined DX, i.e., 'HOV diverge point', where the exit demand of this ramp leaves HOV lane. The locations of these points are internally determined by the simulation module using the procedure described in this section. Figures 5-3 and 5-4 illustrate the interaction between the diamond HOV and normal lanes.

Locating HOV merge and diverge points:

It is assumed that the distance required by HOV demand to enter HOV lane is proportional to the number of lanes on the mainline freeway (HOV lanes not included). It is assumed that each lane change operation requires 200 ft which, incidentally, is equal to twice the DX size in current KRONOS model. Therefore, if there are 'n' lanes in the mainline, the merging traffic needs to go through '2n' DXs before it could merge into HOV lane. This lane changing distance is treated as a parameter in the simulation module and different values can be entered depending on the geometric and traffic conditions. Similar method is adopted for fixing the HOV diverge points upstream of an off ramp. Due to the limitations of modeling, a DX cannot be both a merge and diverge point. If there is a contention for a DX for merge and diverge points, diverging point is given preference and merge point is shifted by one DX forward. Using this method, the merge and diverge points for each ramp are fixed. These DXs are determined at the beginning of the simulation and remain unaltered throughout the simulation run.

Determination of HOV Merge and Diverge Volumes:

The next step after determining the location of merge and diverge points is to determine the amount of flow that can enter/exit to/from HOV at each of these points. Since each on/off ramp has its own merge/diverge location, the demand at each of these points can be considered as the HOV demand on the ramp. The amount of the volume that can enter HOV region at a given DX is determined by the available space in the HOV.

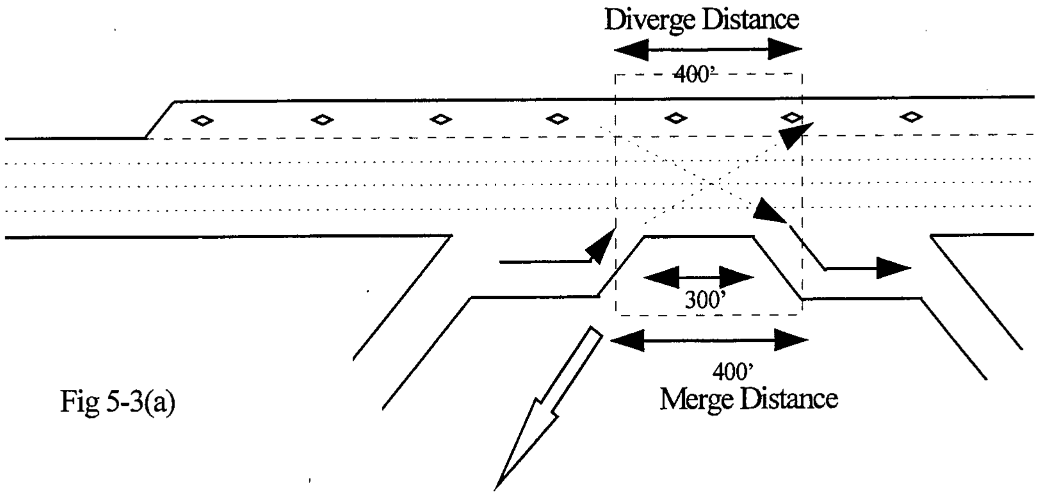
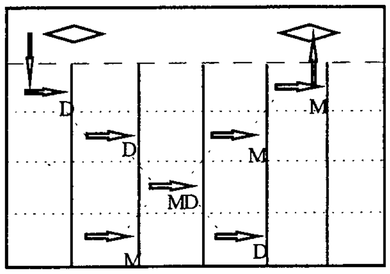


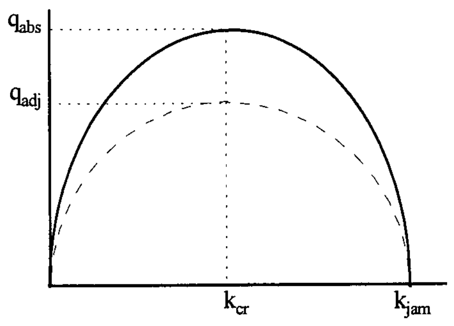
Fig 5-3(a)



M - Merge Volume
D - Diverge Volume
Conflict Volume = M+D

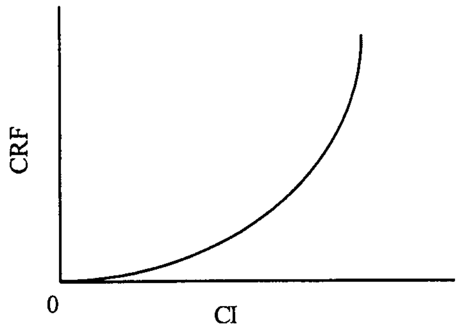
Fig 5-3(b)

Merge/Diverge Distance = K * (#of lanes to cross)
K = 100 ft



q_{abs} = Absolute Capacity of DX
q_{adj} = Adjusted Capacity of DX
= CRF * q_{abs}

Fig. 5-4 Q-K Relationship



CRF - Conflict Reduction Factor
CI = Conflict Index = f($Q_{Through}$, $Q_{conflict}$)

Fig. 5-5 Relationship between Conflict Index Capacity Reduction Factor

This available space is determined as the difference between the jam density and the current density for this DX in HOV region. Similarly, the amount of flow that can exit a HOV at a diverge DX is determined by the available space at that DX in the mainline region. It is assumed that all the HOV entry/exit demand will successfully enter/exit the HOV lane, not necessarily from the point designated as its entry/exit point. If all the HOV demand of an on ramp cannot enter the HOV lane at its designated merge point, the balance flow value is carried forward and is treated as additional entry demand at the next merge point, downstream of the current merge point.

The above process is implemented for all HOV merge points. Similarly, if the diverge demand cannot exit the HOV at a HOV diverge point, the balance flow is extracted from a diverge point upstream of the current diverge point. This process of determining the flow values at each merge/diverge points is carried out at every DT interval (= 1 sec) before the simulation is run for that DT.

The volumes determined in the above step are used while calculating the densities for the next DT using the normal simulation procedure. The HOV merge volume becomes the generation flow for the HOV region and dissipation for mainline region. Similarly, the HOV diverge volume becomes dissipation for HOV region and generation for mainline region.

When mainline region is congested resulting in the failure of entire demand of on ramp to get into the freeway, it becomes difficult to determine the proportion of HOV volume in the demand that entered the freeway. If the traffic volume that enters the freeway is less than the HOV demand, we assume that all the demand that entered the freeway is HOV demand. If it is more than the HOV demand then we assume that all the HOV demand could get onto the freeway and the balance is considered as a portion of the normal demand.

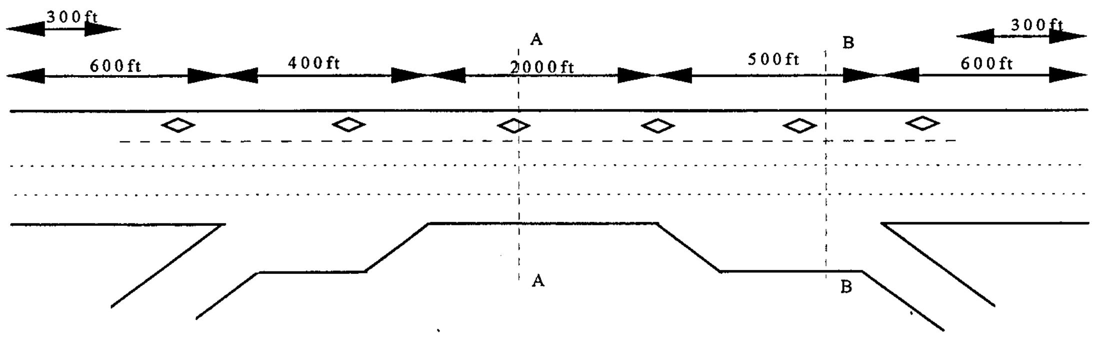
Modeling the effects of Lane Changing

Modeling the effects of the lane changing activities on the traffic parameters has been a challenge to traffic researchers. In this research, it is assumed that the conflicts caused by lane changing affects the flow-density relationship for the affected area. It is further assumed that the effects of the lane changing can be quantified as the reduction of the capacity, the maximum flow at the flow-density curve, for each dx in the lane changing area. This reduction in capacity is a function of the amount of lane changing volume and the amount of through traffic. Using these two quantities, a “Capacity Reduction Factor (CRF)" is developed to determine the amount of capacity reduction because of the traffic conflict. Figures 5-4 and 5-5 illustrate the relationships between CRF and the flow-density curve. The amount of lane-changing volume in a DX is equal to the sum of HOV merge and diverge volumes within that DX. The merging and diverging components in each DX is calculated as follows:

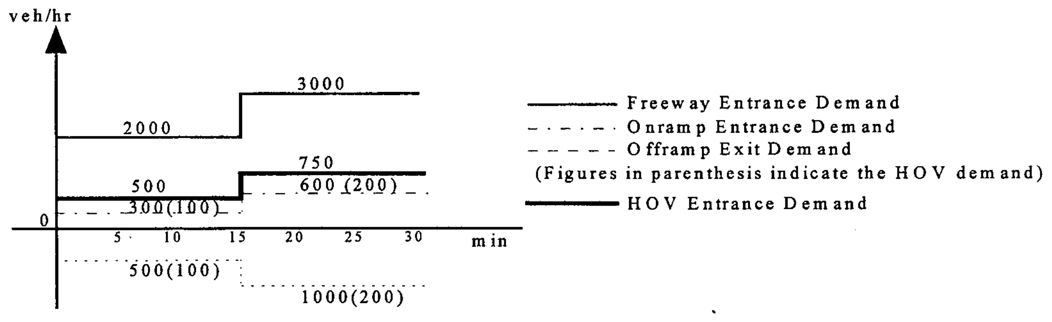
All DXs between a HOV diverge point and its off ramp will have HOV diverge volume and all DXs between an on ramp in HOV region and its HOV merge point will have HOV merge volume. The sum of these two components gives us the total amount of conflicting volume (Fig. 5-3). This volume is computed at every DT (1 sec) interval. This essentially means, that the QK curves of all DXs in HOV region are updated every 1 second.

V. 4 Testing the Diamond HOV lane module

The diamond HOV lane simulation module developed in this chapter was qualitatively tested using an example test section. Figure 5-6 shows the configuration of the test section and the demand pattern through time. The simulation results are summarized in Table 5-1, which also compares the expected values at both the diamond HOV lane and normal lanes. As indicated in the table, the numerical simulation error ranges from 0.8 to 1.5%. Subsequent phase of this research should conduct further testing with real geometrics and HOV lane demand data to calibrate the parameters in the model.



Test Case used for Qualitative Testing
(not drawn to scale)



Demand Pattern used for this Test Section

Figure 5-6 Geometrics and traffic demand pattern for the Diamond HOV lane module

Table 5-1 Test results for the Diamond HOV lane simulation module

Time in Minutes	HOV Lane Flow at A-A (veh/hr)			Mainline Flow at A-A (veh/hr)		
	Expected	Simulated	%Error	Expected	Simulated	%Error
5	600	591	1.5	1700	1682	1.05
10	600	591	1.5	1700	1682	1.05
15	600	591	1.5	1700	1682	1.05
20	950	937	1.4	2800	2776	0.85
25	950	937	1.4	2800	2776	0.85
30	950	937	1.4	2800	2776	0.85

VI. DEVELOPMENT OF A NEW SIMULATION MODULE FOR TRAFFIC RESPONSIVE, AUTOMATIC RATE-SELECTION RAMP METERING STRATEGIES

VI. 1 Overview of the Mn/DOT traffic-responsive ramp metering scheme

By regulating the volume entering mainline from on-ramps depending on the traffic conditions in real time, the Mn/DOT metering algorithm seeks to prevent the congestion and reduce the accidents. It has been proven that, with the Minnesota algorithm, a sustained flow rate of 2200 to 2400 vehicles per lane per hour can be achieved for hours (Lau, 1996). This section overviews the basic concept of the Minnesota algorithm that is based on "metering zone" and "bottleneck"

Metering zone

A metering zone is defined as one direction of freeway typically three to six miles in length. The beginning or upstream end of a zone is usually a free-flow area not subject to a high incident rate. Within a zone there are several metered entrance ramps, a number of exit ramps, and perhaps one or more non-metered entrance ramps. The downstream end of a zone is at a critical bottleneck, where the demand to capacity ratio is highest on that freeway zone. The usual types of bottlenecks include lane-drop locations, major volume entrance ramp areas and major volume weaving areas.

Zone-based metering algorithm (Mn/DOT 80)

The metering algorithm that has been implemented since early 1980s is built on the basic concept of equalizing the incoming traffic volumes with those traffic volumes leaving each zone. The basic equation can be expressed as:

$$A + U + M + F == X + B \qquad \text{(Eqn 6-1)}$$

where,

A = Upstream mainline volume (measured variable)

U = Non-metered entrance volumes (measured variable)

M = Metered local access ramps volume (sum of controlled variables)

F = Metered freeway to freeway ramps volume (sum of controlled variables)

X = Exit ramp volumes (sum of measured variables)

B = Downstream bottleneck capacity volume (constant)

Equation 6-1 can be rewritten as

$M + F = X + B - A - U$ (Eqn 6-2)

The above equation indicates that any measured variation in (X + B - A - U) is equaled by a controlled variation in (M + F). Therefore, the objective of the metering scheme is to determine the appropriate values for (M + F) for the measured value of (X+B-A-U) for each metering time interval.

Target Volumes

Using the historical data measured from the detectors, the balanced form of Eqn 6-2 can be

derived for each zone after proportional minor adjustments. The values in the balanced equation are called as target volumes and used as default values when the measurements are not available.

Metering rates

Each local access metered ramp is assigned six metering rates, which would equate to 1.5, 1.3, 1.1, 0.9, 0.7 and 0.5 times the target volume for that ramp over a five-minute time. On freeway to freeway ramps, the rates over a five-minute time would be 1.25, 1.15, 1.05, 0.95, 0.85 and 0.75 times the target volume. The selection of which rate to use is based on a comparison of the measured variables (X+B-A-U) to a series of thresholds as follows;

(X+B-A-U) > Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
1.4M +1..2F	1.2M +1..1F	1.0M +1..02F	0.8M +0.9F	0.6M +0.8F	

The six rates are achieved by the use of six red times with the fixed yellow and green times of 1.3 and 0.7 seconds.

Occupancy control

The above algorithm depends on the traffic conditions at each zone bottleneck, i.e., the location of predefined bottlenecks remain as the controlling locations. However, during unusual traffic conditions or any incident periods, other locations may become controlling bottlenecks. In this situation, the occupancy values measured every 30 second increments across all lanes in a station are used to determine the metering rates for each ramp. Each metered ramp is assigned detector stations up to three miles downstream for occupancy control. The highest occupancy measured for one minute at any station is compared to the following series of occupancy thresholds;

Measured Occupancy value	Rate level
< 17%	3
18	4
23	5
> 40	6

If the rate selected based on occupancy thresholds is more restrictive than the volume control rate, then it will be used for the next 30-second interval.

Turn on/off metering thresholds

The AM peak has a turn on period (6:00a.m. - 7:00a.m.) during which ramp meters will turn on when a restrictive rate 5 or 6 is called for on three consecutive 30 second intervals. During the turn off time period (8:00a.m. - 9:30a.m.), a ramp meter will turn off when the arrival rate falls and the ramp empties. Every 5 minutes, the ramp volume recorded downstream from the meter signal is compared to the number of greens displayed during that 5-minute interval. When the measured volume falls below 90% of the number of greens, then the meter is turned off. During the ensured metering period, i.e., 7:00a.m. - 8:00a.m., all ramps are metered. The PM peak has also three parts: the turn-on period (2:00p.m. - 3:30p.m.), the ensured metering period (3:30pm. - 5:30p.m.) and turn off period (5:30p.m. -7:00p.m.).

VI. 2 Development of a simulation module for Mn/DOT 80 metering algorithm

In this section, a new module was developed to simulate the Mn/DOT 80, traffic responsive, automatic rate-selection ramp metering strategy.

Modeling traffic sensors

In order to provide on-line ramp metering, it is necessary to implement traffic sensors. The basic detector type in KRONOS is

```
typedef struct {

  int    Detector_ID,
            Type,
            Lane_Num,      /* The lane that the detector is in,
                          * numbered from the right most lane */
            Offset,        /* The distance in feet from the head of
                          * the segment to the detector */
         STATUS;           /* Is the detector producing_valid_data,
                          * producing_invalid_data or
                          * not_producing_data */

  boolean    ON_MAINLINE;        /* Is this detector on a ramp or
                          * on the freeway mainline? */

  unsigned int Outputs,          /* bit field for outputs available from
                                  * detector */
                 Valid;          /* bit field for valid data in detector
                                  * data file */

  int          Lane_Num[MAX_NUM_OUTPUT_TYPES],
                    /* The lane that the detector is in,
                     * numbered from the right most lane */
            OffsetMAX_NUM_OUTPUT_TYPES,
                            /* The distance in feet from the head of
                             * the segment to the detector */
               Num_Lanes[MAX_NUM_OUTPUT_TYPES],
                            /* number of lanes occupied by data type */
               Num_DXs[OUTPUT_TYPES];      /* number of dxs covered by data type */

   /* conversion factor for each data type */
   float        Conv_Factor[MAX_NUM_OUTPUT_TYPES];

   /* pointer to the parent segment which this detector lies in */
   Segment_Ptr  Seg;

   /* array of pointers to dxs */
   Sim_Data_Ptr       Sim_Data[MAX_NUM_OUTPUT_TYPES];

   Station_Ptr  Station;
}
```

Modeling Detector

A detector is responsible for reporting the appropriate data from the correct location on the freeway. A detector can produce up to 16 different types of data, and each data type can correspond to multiple contiguous lanes and contiguous dxs. This is a crucial element of the detector design flexibility. A pneumatic tube counter sums volume across all lanes, but at a single location. Assume that volume has predefined index of 3, and that the segment in question has 5 lanes. For a section with a pneumatic tube counter, the volume bit, bit 3, of Detector.Outputs would be set to 1. (Since this is a single integer being used as a "bit array", and each bit corresponds to a power of 2, the third bit would equal decimal 8. This would be the actual valued stored in Detector.Output). The Detector.Num_Lanes[3] would be set to the number of lanes covered by the pneumatic tube, 5, and the Detector.Num_DXs[3] would be set to 1, since a pneumatic tube would fall within a single DX.

A video detector, such as Autoscope, poses a different variation on the same problem. If a video detector produced volume data over the same location as the tube counter, but only produced density data for the 3 mainline lanes, this could be handled as well. Assume that density corresponds to index 9. Detector.Output would be set to 2^9 + 2^3 (setting each bit--one for volume and one for density). Detector.Num_Lanes[3] would still be 5, and Detector.Num_DXs[3] would still be 1. Detector.Num_Lanes[9] would be set to 3.

In this fashion, any detector type can be represented. A detector can cover many lanes and many dxs. The current version supports loop detectors which cover one lane and one dx. Other types can be implemented as necessary.

The Detector structure has a pointer to a segment, the segment which contains this detector, so that the segment type and number of lanes can be determined by accessing the segment structure--rather than stored again in the detector structure. In addition, the detector has pointers that point directly to the DX of the corresponding data arrays. These pointers are set once when the detector data is read in, eliminating the need to re-compute the address for each DT.

The last hurdle to getting the actual data values requires that the array representation be "undone." In most cases, the simulation data arrays cover more than one lane. It is necessary to determine the number of lanes corresponding to this segment, and then to compute the number of lanes worth of data stored in the array based on the segment type. The array data--flow and density--is then divided by the number of lanes covered by this array to produce the single lane data corresponding to this detector.

Once the simulated detector value is found, the Station pointer is checked to see if it is NULL. If it is not NULL, then this detector participates in a station relation, and the appropriate station aggregation must be performed. If the station's Time_To_Report is true, then the station report's its results to the parent meters which use it. The meter then computes the new rate based on the station results. Each station has a Report_Counter which is decrement each second. When the Report_Counter is zero, Time_To_Report is set to TRUE. In addition, each station has its own Sum_Secs_In_Agg_Period. Even though the input module sets the value once for all stations, future modeling may not take such a simple approach. The station also has a Data array, with the current aggregated value for each of the detector output data types.

```
typedef struct {
  Station_ID     Station_ID;
  Detector_Array  Detectors; /* The array of detectors in this station */
  int            Num_Detectors,              /* the number of detectors in this
                                          * station */
                 Agg_Counter,          /* the counter which tracks time
                                          * until next report */
                 Num_Secs_In_Agg_Period;  /* the length of one aggregation period
                                          * in seconds */
  /* the accumulated values for each data type */
  float Data[MAX_NUM_OUTPUT_TYPES];
  boolean        Time_To_Report;
}           Station;
```

Defining Zone structure

A Zone can be defined with an upstream station, a downstream bottleneck station, a set of volume stations, and a set of volume thresholds. The Zone structure is defined below.

```
typedef struct {
  Zone_ID       Zone_ID;
  Station_Ptr   Upstream_Station,    /* pointer to the station which
                              * counts zone input flow */
                Bottleneck_Station; /* pointer to the station which
                              * counts bottleneck volume */
  int           Num_Volume_Stations,
                Num_Volume_Thresholds;
  Station_Array   Volume_Station;

  Threshold_Array Volume_Thresholds;   /* array of the volume thresholds
```

```
                                * for the meters in this zone */
    }           Zone;
```

Design of Fixed time metering

The simplest meter type is the fixed rate meter. A fixed rate meter has only a Location and a Rate.

```
typedef struct {
        float    Rate;
        int      Location;
}       Fixed_Rate;
```

Automatic rate-selection metering : Mn/DOT 80 algorithm

A ramp meter with the Mn/DOT 80 algorithm and Zone control can be defined as follows:

```
typedef struct {
  Zone_Ptr      Zone;       /* the zone of which this meter is a part */
  int           Num_Rates_In_Table,
               Num_Occupancy_Stations,
               Algorithm,   /* The algorithm used if it is an ON_LINE
                        * meter */
               Num_Operating_Periods;

  Rate_Selection_Table ARS_Table;      /* and this is the actual rate */
  Station_Array   Occupancy_Station;
}        Mn/DOT_Using_Zones;
```

Automatic rate-selection metering without zone control

A third type of meter would use the Mn/DOT80 algorithm, but without Zone control. It is defined as follows.

```
typedef struct {
  Station_Array   Volume_Station,
                  Occupancy_Station;
  int             Num_Rates_In_Table,
                  Algorithm, /* The algorithm used if it is an ON_LINE
                             * meter */
                  Num_Occupancy_Stations,
                  Num_Volume_Stations,
                  Num_Operating_Periods;

  Rate_Selection_Table ARS_Table; /* and this is the actual rate
                                   * selection table */
}              Mn/DOT_Not_Using_Zones;
```

Finally, a Ramp_Meter is defined as

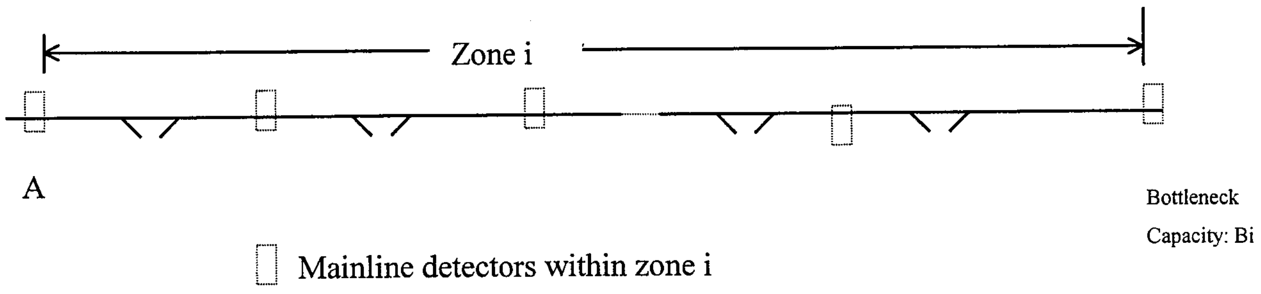
```
typedef struct {
  boolean        USING_ZONE_CONTROL;  /* is this meter operating under
                                      * zone control? */
  int            Location,    /* keep track of the number of detectors */
                 Type;        /* The class of meter ( FIXED, ON_LINE ) */
  Segment_Ptr    Seg;
  float          Rate;
  /* allocate storage for only one of these 4 types */
  union {
    Fixed_Rate    Fixed;
    MNDOT_Using_Zones MN_UZ;
    MNDOT_Not_Using_Zones MN_NUZ;
  }           Class;
}           Ramp_Meter;
```

As described above, in this research, a generic meter data type is defined in the Ramp_Meter structure, and the ramp specific data is stored in a separate structure. The union operator simply allocates enough memory for the largest of the elements listed within it. In this way, new structures for new meter types can be designed, and simply included in the union in Ramp_Meter.

The above representation of the meter data structure has two advantages. First, it is clear by inspection which data is needed for each meter type. Second, instead of having many different arrays of each meter type, the union provides a mechanism for putting data of any meter type in the same place, which allows uniform access to all meters regardless of type.

VI. 3 Overview of new metering algorithm: Mn/DOT 97

In this section, the modified metering algorithm, called Mn/DOT 97 and currently being tested in the field by the Traffic Management Center, is described. The Mn/DOT 97 algorithm, developed by the TMC staff, will be incorporated into the simulation module at the subsequent phase. The new algorithm is based on the same zone metering concept as the existing Mn/DOT 80 algorithm, but introduces the new concept of available space at each zone in determining the metering rates and uses the exit volume data measured in real time. Further, it has the capability to use the variable capacity values at the bottleneck locations.



Let Mt : Sum of Metering rates for all the ramps metered during time interval t within zone i

Ft : Sum of Freeway to Freeway metering rates during t within zone i

Ut : Sum of entering volume from all non-metered ramps during t within zone i

Xt : Sum of Exit ramp volume for all the exit ramps during t within zone i

At: Upstream boundary measured volume during t

Bi : Capacity of bottleneck B of zone i

As the previous metering logic,

1) each ramp has 6 predefined rates, i.e., red times, and volume and occupancy thresholds.
2) all the metered ramps within a zone have same volume thresholds and volume detector station, i.e., station A (upstream boundary detector station)
3) every 30 second, two rates are selected with volume thresholds and occupancy thresholds using the measurements.
4) Occupancy control is same as the previous one.

New metering algorithm based on volume measurements

Every 30 second, the system calculates the Vi value with the following measurements:

$$Vi = Bi - At - Ut + Xt + St$$

where, St = Sum of all mainline detector within zone i for the following quantity

$$= \sum (Od - Oc_t)_l \ * \ alpha_l * (\text{number of lanes at l})$$

Od_l = Critical occupancy for detector l (Input parameter for detector l)

Oc_t = Occupancy measurements from detector l during t interval

$alpha_l$= Conversion parameter for detector l (input parameter : default value 1.2)

Bi = Bottleneck capacity (Input parameter for zone boundary detector)

All the values in the above equation is 5-minute values. However,

At, St => 5-min converted volume using 1 minute volume measurement. At is used for next 30 seconds.

Ut, Xt => 5-min measurements used for the next 5-minute interval.

The above Vi goes to the volume thresholds table and one rate is selected.

As with the previous algorithm, the two rates, i.e., volume-based and occupancy-based rates, are compared every 30-second and the more restrictive rate is selected for the next 30 second implementation. Further, the volume thresholds for each zone are determined following the same procedure as with the previous algorithm.

Turn on condition: Every 5 minutes, measure M and F, i.e., sum of total entered volume from metered ramps and if

$$(M+F)_i > beta *(B - A - U + X + S)_i,$$

then turn on all the meters in zone i. beta = input parameter (Default 0.85)

Turn off condition: For each ramp,

Actual volume entered for last 5-min. interval <

gamma * (total metering rate for last 5 minute interval),

then turn it off.

where, gamma = input parameter. (Default value: 0.85).

VII. DEVELOPMENT OF WINDOWS-BASED USER INTERFACE AND DEMAND DATA LOADING PROCEDURE

VII.1 Overview of windows-based user interface

Developing a user-friendly environment where the user can easily enter the data needed for simulation and analyze the output results is of critical importance for practicing engineers in applying simulation for effective traffic management. The user interface of the previous version, developed under the MS-DOS environment, has introduced the new concept of graphical, interactive data input procedure, which has enabled a user to build a freeway on the screen by combining segment icons with a mouse. While the user interface of the previous version has many desirable features, the initial effort to enhance it to support new simulation features, such as ramp metering, network and HOV lanes, has turned out to be not feasible because of the limitations of the DOS-based graphical programming environment. In consultation with the Minnesota Department of Transportation, it was decided to develop a windows-based user interface and the Borland Delphi software package has been selected as the development tool kit for the windows-based user interface. Delphi is one of the most powerful, software development tools for the Windows environment currently available for the PC platform. It combines visual development with a complete OOP language and allows full access to the underling Windows API system in a user-friendly way. Using Delphi, it only requires small amount of code to develop an application that requires hundreds or even thousands of lines with C or assembly languages. Further, the flexible environment of Windows makes it possible to combine the user interface with other heterogeneous programs such as C , C++, HTML, database and GIS software. The rest of this chapter summarizes the features and the data loading procedure with the newly developed user interface.

VII. 2 Design and implementation of input/output screens with Delphi

The design of the new interface is based on the pop-up menu concept common to Windows-based applications. However, the data input screens use the multiple hidden page structure, which makes it possible to enter all the data without exiting the main screen. Figure 7-1 shows the main geometry input screen where user can build a freeway by selecting appropriate segment icons in sequence from the toolbox. Currently 20 segment types are available and there is no limitation in terms of the number of lanes in the on-ramp as long as the total mainline lanes do not exceed eight. Further, the user can specify the mile point as well as the number of lanes for the first segment.

Global parameter input screen

The global parameters can be entered using "Edit parameter" menu in the main screen. Figure 7-2 shows the sub-window for the global q-k curve data in the Edit Parameter window. Figure 7-3 also shows the volume/occupancy threshold data window, which allows the user to save a particular set of thresholds in the files that can be used for the ramp metering simulation. The other sub-windows include Simulation Parameters to input simulation control data, Boundary Conditions for defining the traffic demand data conditions at upstream and downstream boundaries, and Metering Options to specify ramp metering control schemes to be simulated. Currently, user can select one of the three control schemes, i.e., non-metering, Fixed-time metering, and automatic rate-selection metering. The default option is non-metering and other data screens will show up when any of the other two options is selected.

Segment data input screen

All the segment specific data can be entered in the segment data input window, which can be popped up by double clicking each segment . Figure 7-4 shows the geometry data input sub-window for a weaving section. As indicated in the figure, all the other data, such as demand and detector location, can be entered without exiting the main screen. For example, Figure 7-5 shows the input screen for the automatic rate selection ramp metering data.

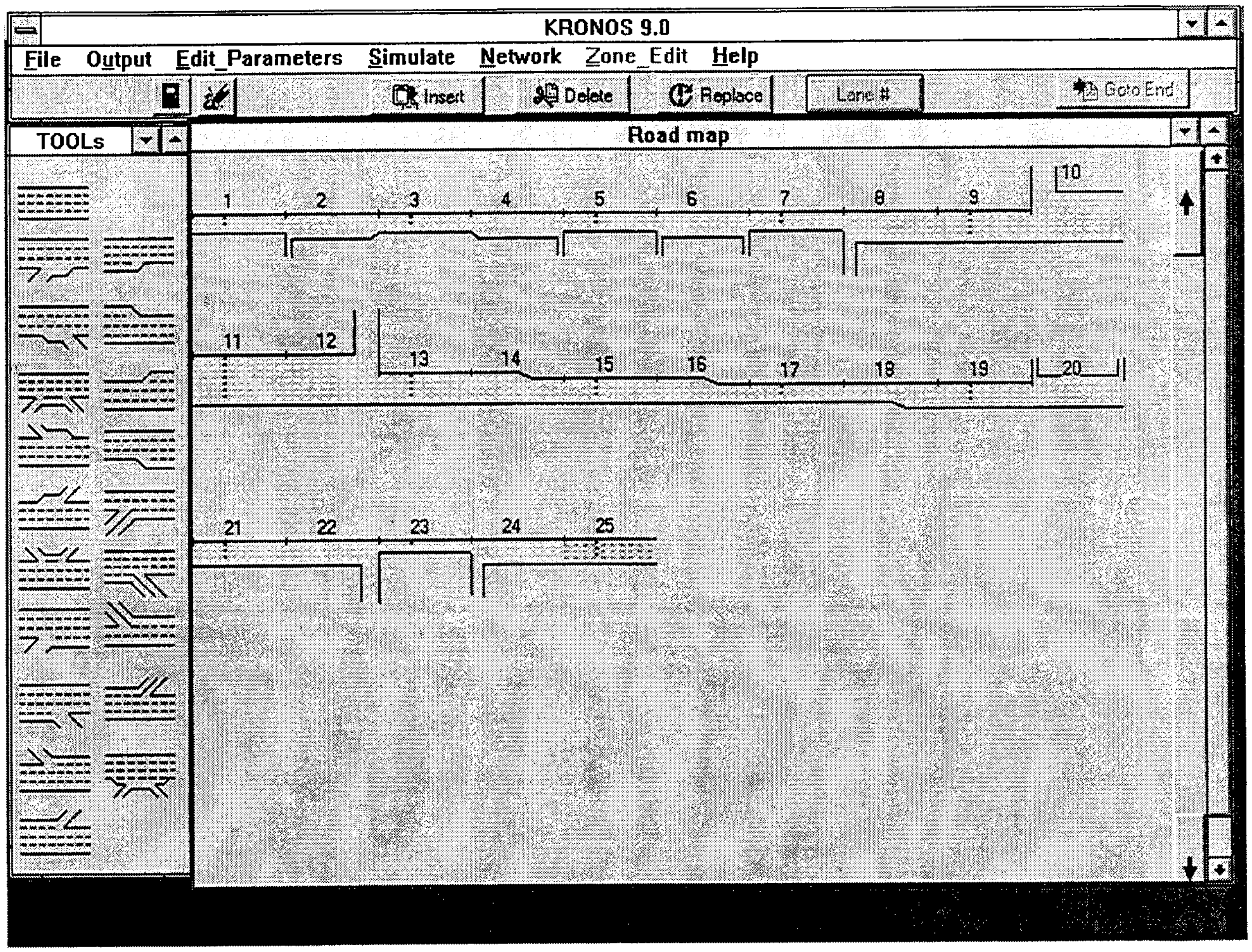
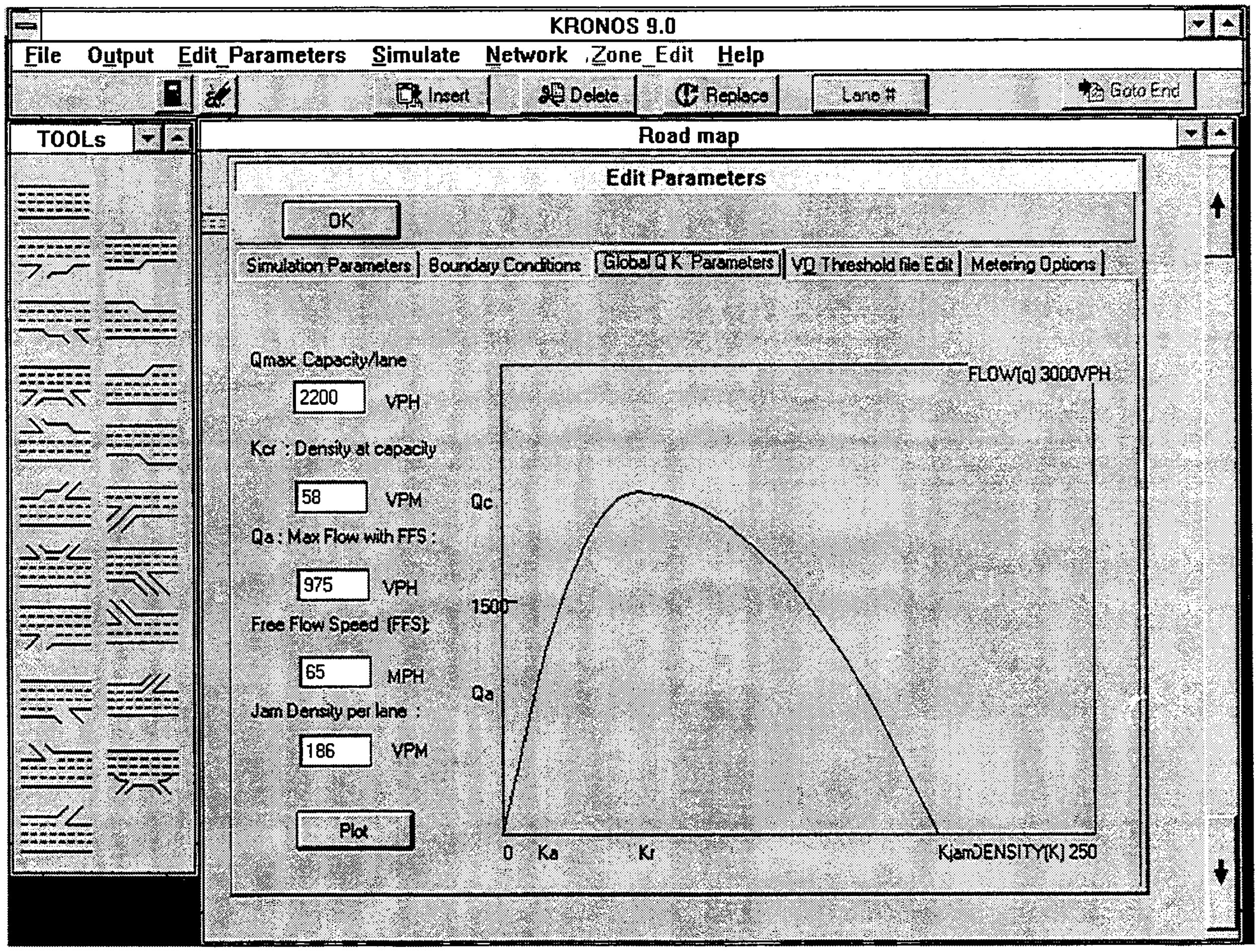
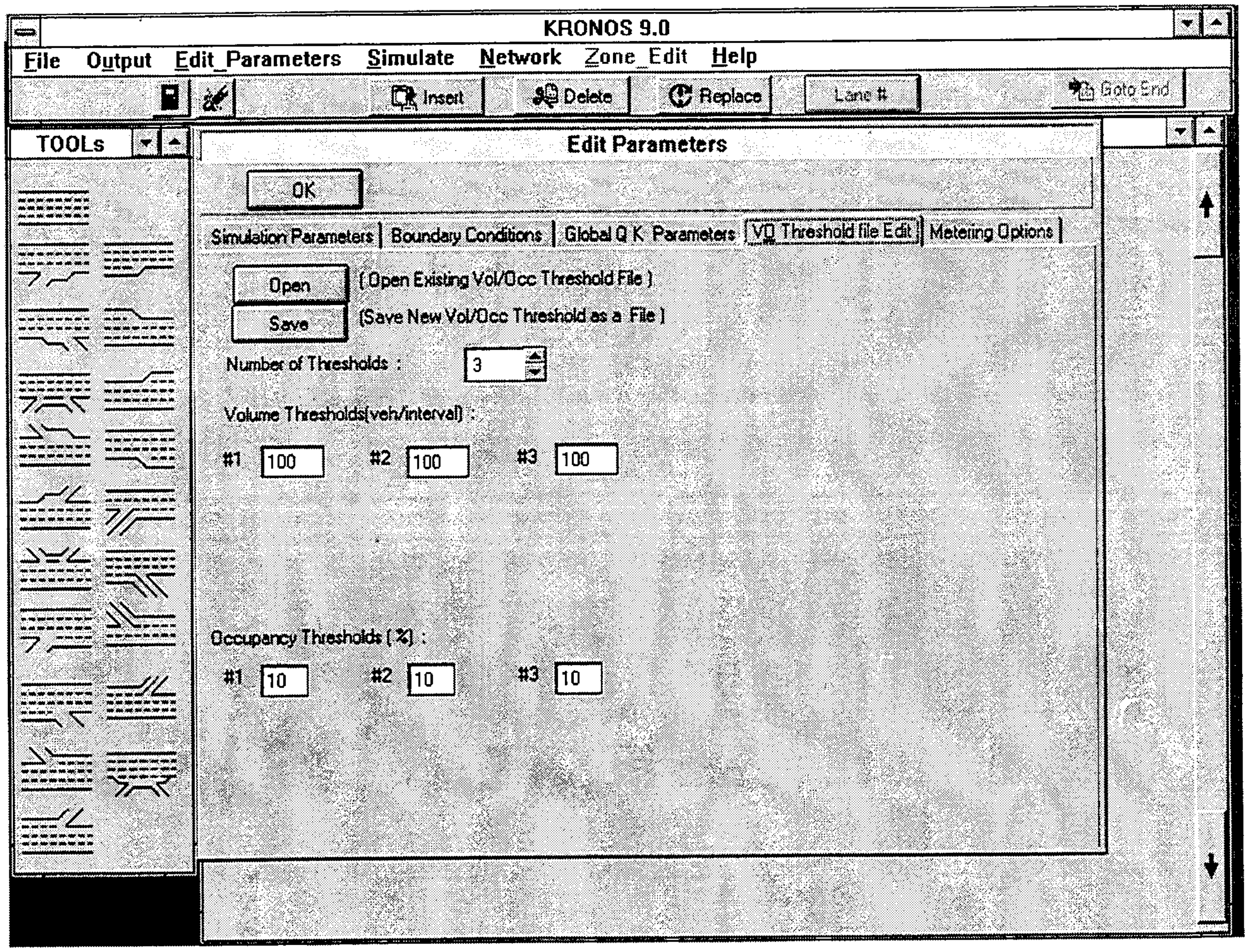


Figure 7-1 Freeway geometry data input window



Qmax: Capacity/lane
2200 VPH
Kcr : Density at capacity
58 VPM
Qa : Max Flow with FFS :
975 VPH
Free Flow Speed (FFS):
65 MPH
Jam Density per lane :
186 VPM

Figure 7-2 Global flow-density relationship data input window



(Open Existing Vol/Occ Threshold File)
(Save New Vol/Occ Threshold as a File)

Number of Thresholds : 3

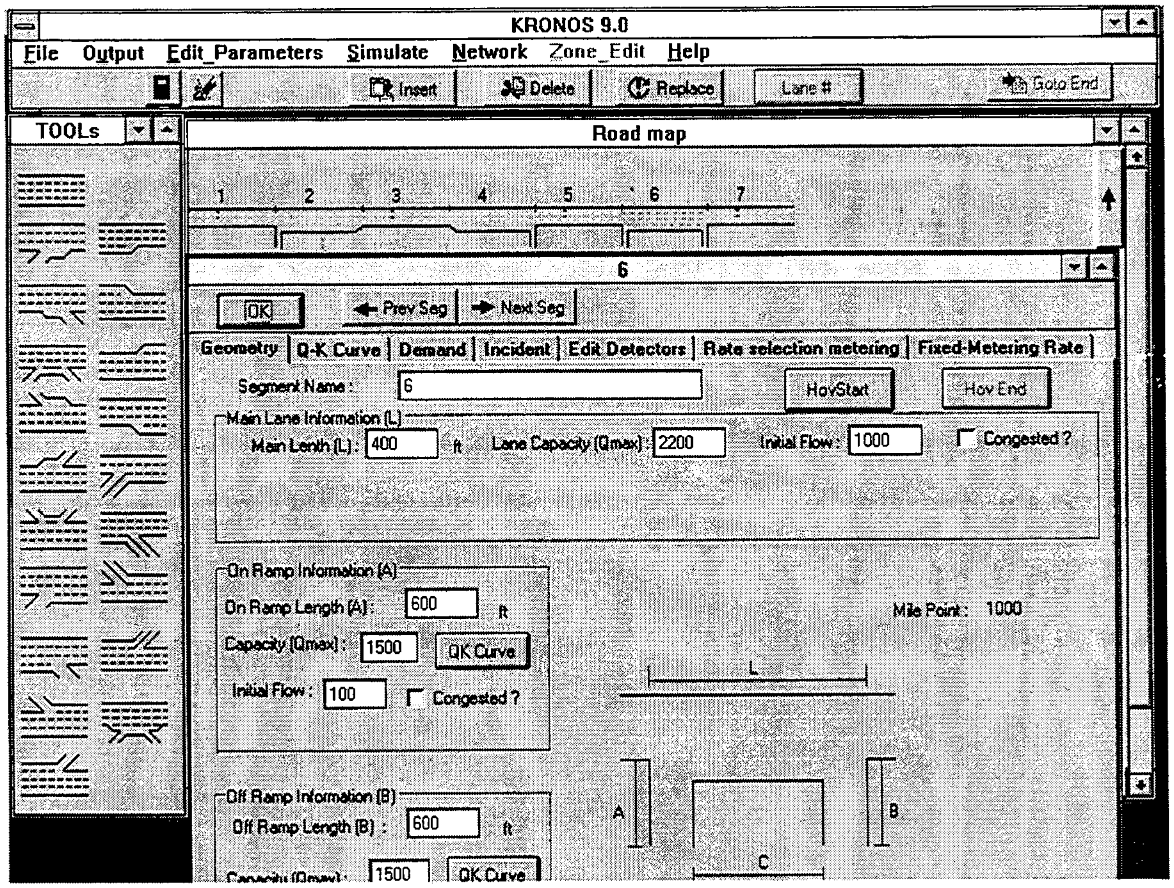
Volume Thresholds(veh/interval) :

#1 100 #2 100 #3 100

Occupancy Thresholds (%) :

#1 10 #2 10 #3 10

Figure 7-3 Volume/Occupancy threshold file data input window



Segment Name : 6

Main Lenth (L) : 400 ft Lane Capacity (Qmax) : 2200 Initial Flow : 1000 ☐ Congested ?

On Ramp Length (A) : 600 ft

Capacity (Qmax) : 1500

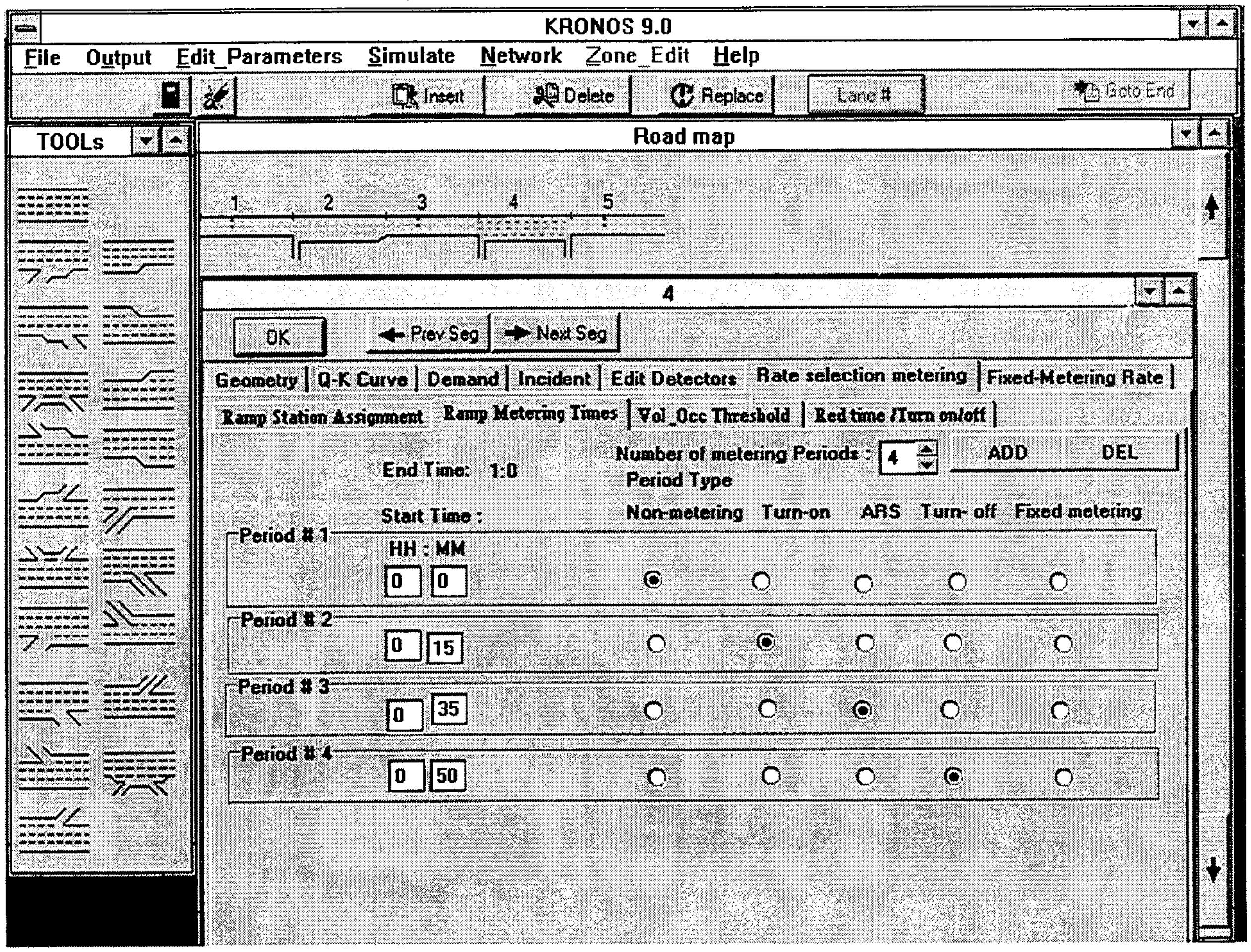
Initial Flow : 100 ☐ Congested ?

Mile Point : 1000

Off Ramp Length (B) : 600 ft

Capacity (Qmax) : 1500

Figure 7-4 Example segment data input window



End Time: 1:0

Number of metering Periods : 4

	Start Time : HH	MM	Non-metering	Turn-on	ARS	Turn- off	Fixed metering
Period # 1	0	0	●	○	○	○	○
Period # 2	0	15	○	●	○	○	○
Period # 3	0	35	○	○	●	○	○
Period # 4	0	50	○	○	○	●	○

Figure 7-5 Example metering periods input window

Output data screen

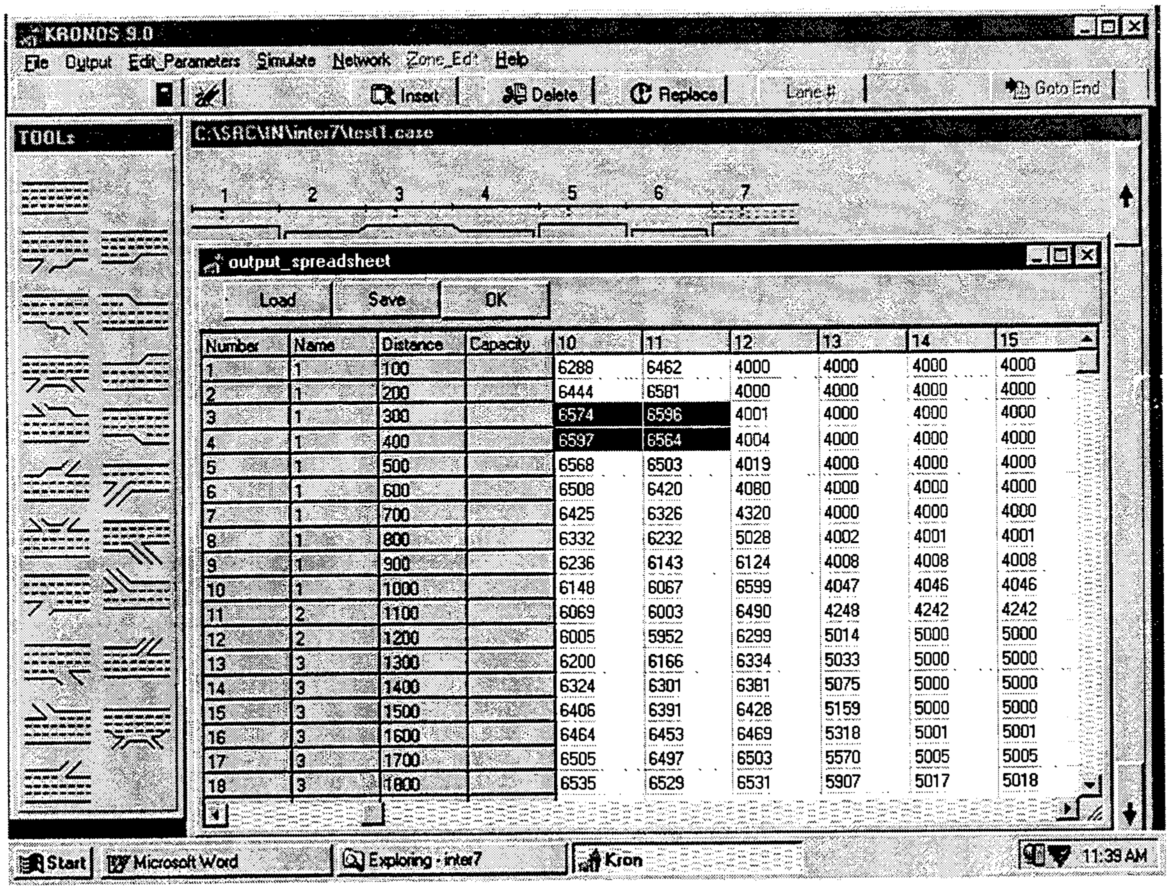
There are three types of output in terms of format: spreadsheet, contour and 2-D graphics. Figure 7-6 shows an example total flow spreadsheet output window, which includes the total flow value for every 100ft dx for every output aggregation time interval specified by user. Currently, the following output results are available in the spreadsheet format:

Total flow
Average lane flow for each time interval
Average lane density for each time interval
Average lane speed for each time interval
Instantaneous average lane flow
Instantaneous average lane speed
Instantaneous average lane density

The contour output screen shows the variation of an output parameter in the time and distance space using the continuous color scale. Figure 7-7 shows an example contour screen for the average lane flow. User can also find out the value of the output parameter in the contour plot for a specific time and distance by moving the lever on the time and distance axis. Figure 7-7A shows the available output parameters for the contour plot. Those include

average flow per lane
average density per lane
average speed per lane
total flow
total delay
average delay per lane

The two-dimensional plot can be drawn for an output parameter selected for the contour plot. Figure 7-8 includes an example two-dimensional plot for the average lane flow.



Number	Name	Distance	Capacity	10	11	12	13	14	15
1	1	100		6288	6462	4000	4000	4000	4000
2	1	200		6444	6581	4000	4000	4000	4000
3	1	300		6574	6596	4001	4000	4000	4000
4	1	400		6597	6564	4004	4000	4000	4000
5	1	500		6568	6503	4019	4000	4000	4000
6	1	600		6508	6420	4080	4000	4000	4000
7	1	700		6425	6326	4320	4000	4000	4000
8	1	800		6332	6232	5028	4002	4001	4001
9	1	900		6236	6143	6124	4008	4008	4008
10	1	1000		6148	6067	6599	4047	4046	4046
11	2	1100		6069	6003	6490	4248	4242	4242
12	2	1200		6005	5952	6299	5014	5000	5000
13	3	1300		6200	6166	6334	5033	5000	5000
14	3	1400		6324	6301	6381	5075	5000	5000
15	3	1500		6406	6391	6428	5159	5000	5000
16	3	1600		6464	6453	6469	5318	5001	5001
17	3	1700		6505	6497	6503	5570	5005	5005
18	3	1800		6535	6529	6531	5907	5017	5018

Figure 7-6 Example spreadsheet output window for total flow

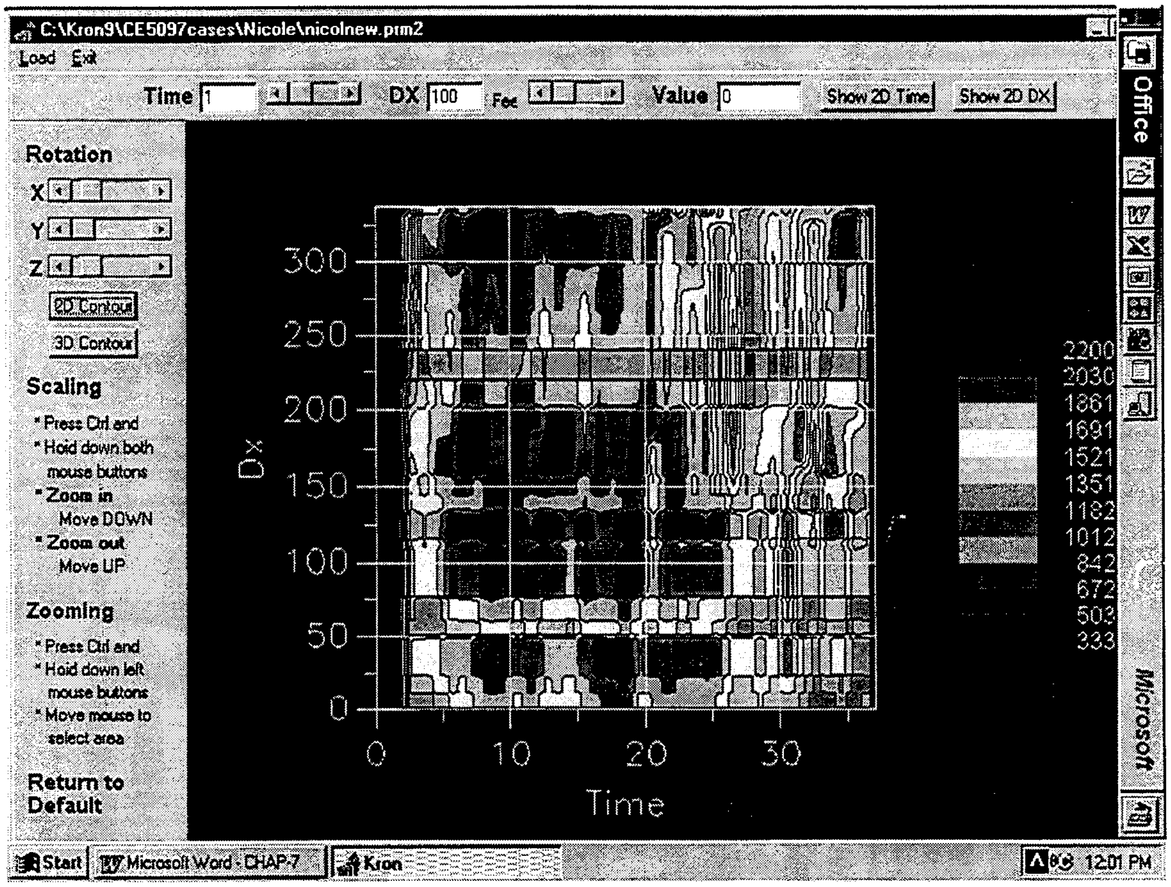
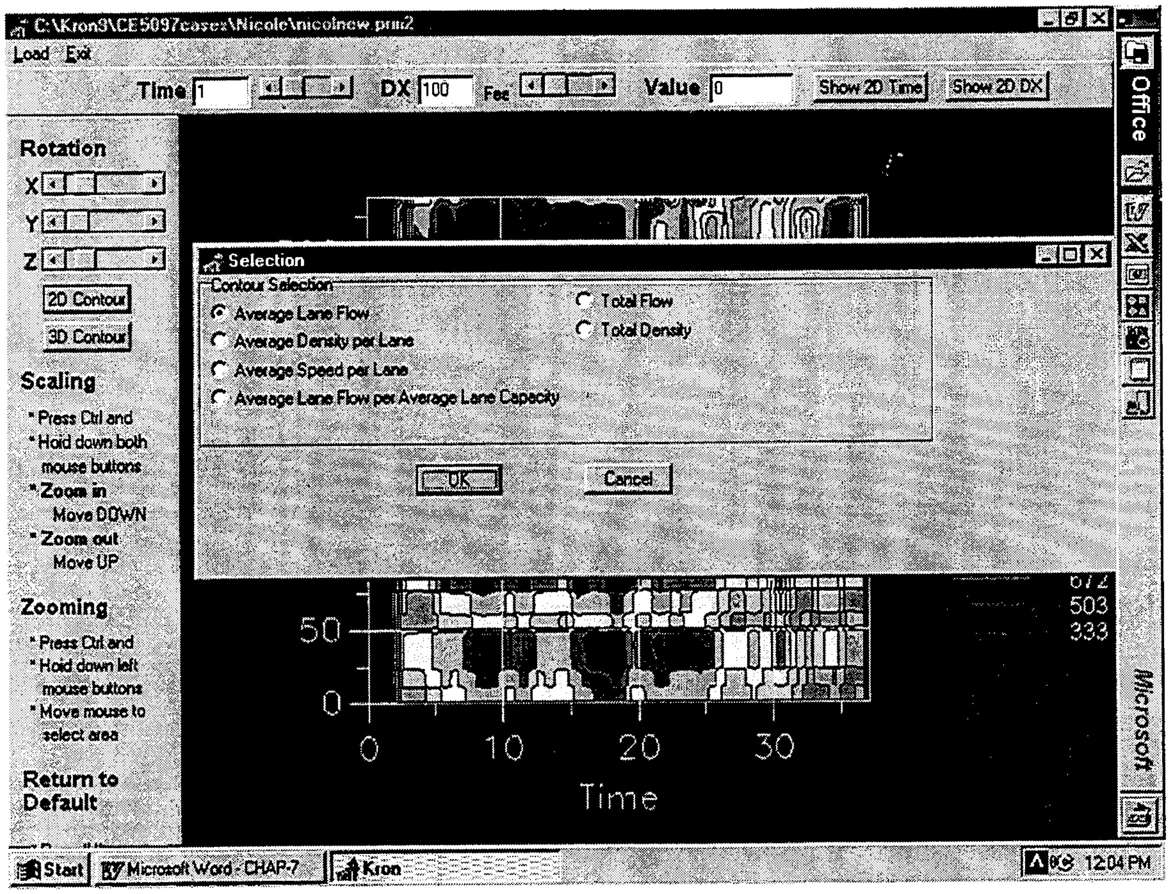


Figure 7-7 Example output contour screen



Scaling
* Press Ctrl and
* Hold down both mouse buttons
* Zoom in Move DOWN
* Zoom out Move UP

Zooming
* Press Ctrl and
* Hold down left mouse buttons
* Move mouse to select area

Contour Selection
[✓ Average Lane Flow]
Average Density per Lane
Average Speed per Lane
Average Lane Flow per Average Lane Capacity
Total Flow
Total Density

Figure 7-7a. Available options for contour output

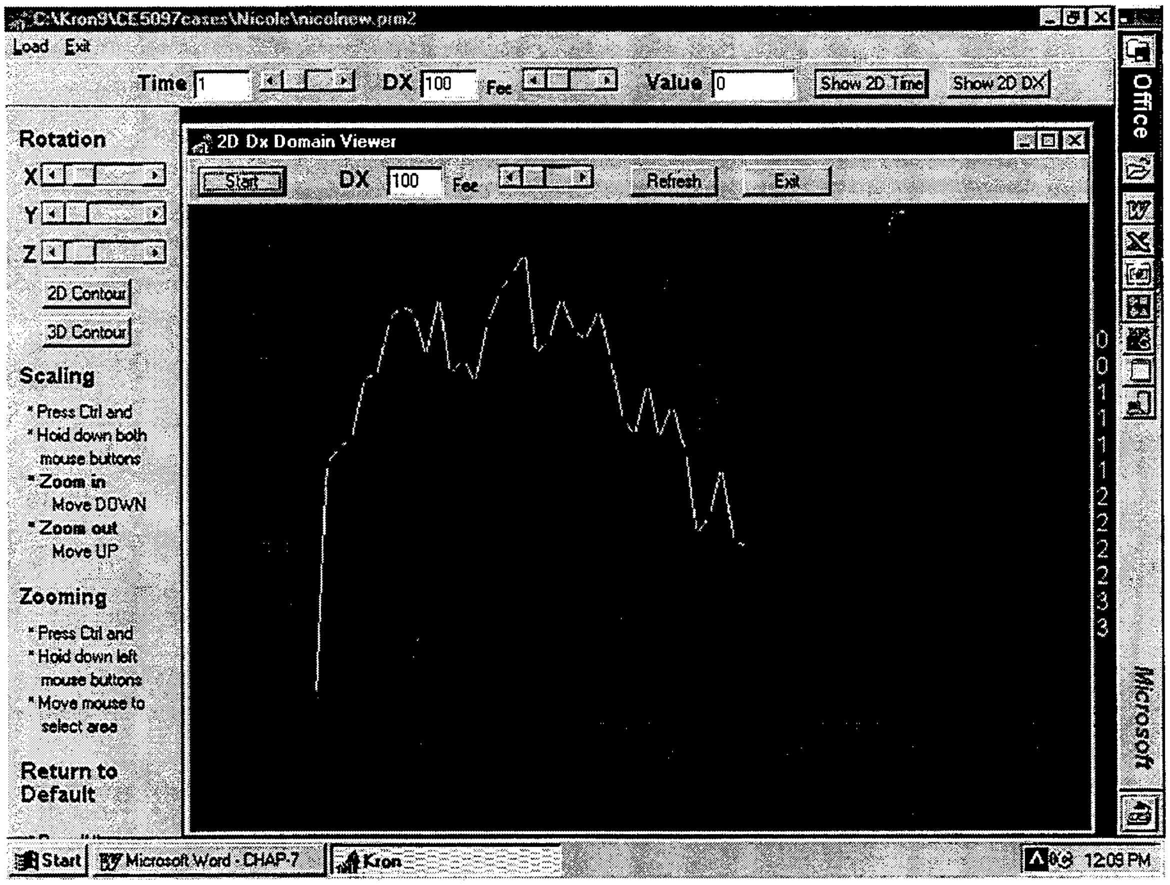


Figure 7-8. Example 2-D graphical output screen

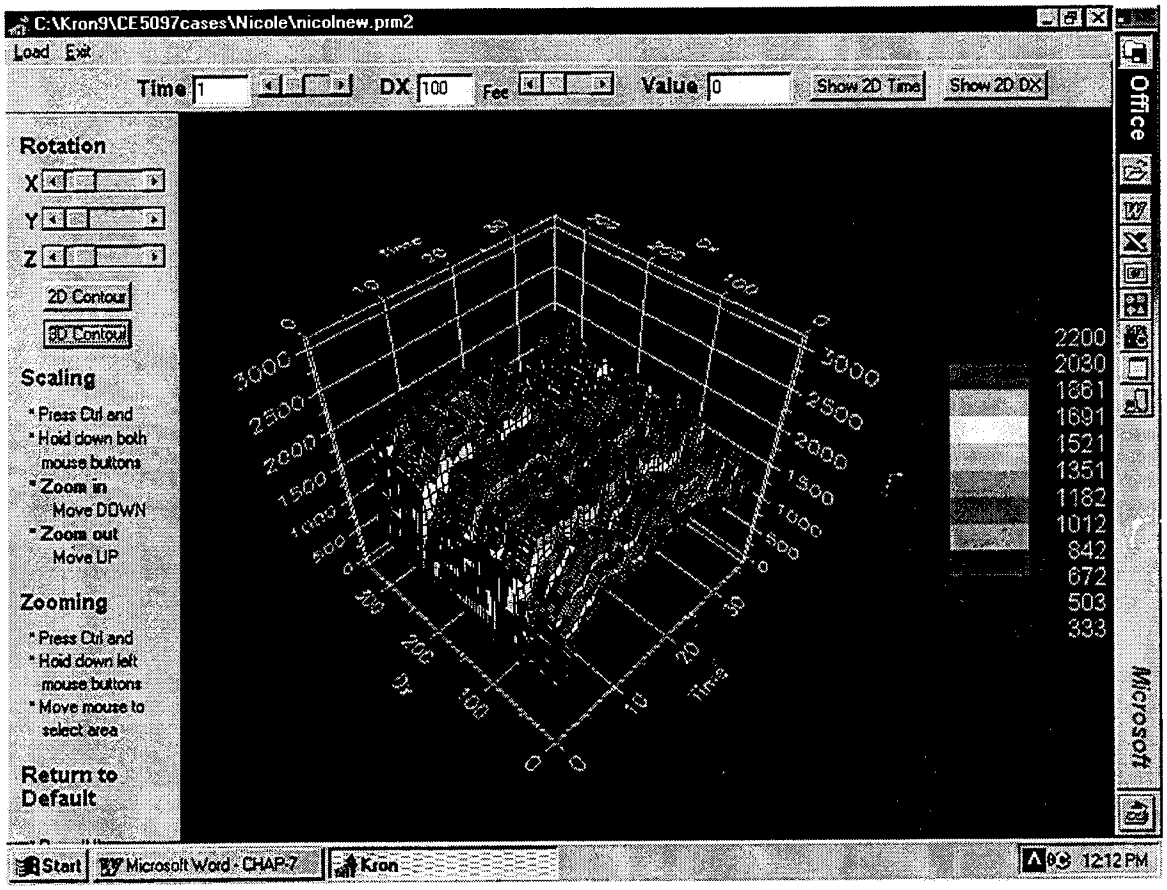


Figure 7-8a Example 3-D graphical output screen

VII.3 Development of automatic demand data loading procedure

Traffic demand data at the external boundaries for a given freeway section is one of the key input data absolutely necessary for simulation. The external boundaries include all the entrance/exit ramps and upstream as well as downstream segments. In this research, an automatic demand data loading procedure is developed when user wants to use the detector readings at each boundary as the demand data. The detector readings are currently stored in a binary file for each day by the Traffic Management Center. While the detector data was supposed to be stored in a traffic database, being developed in a separate project, the status of the traffic database development has not reached the point where it's possible to interact with an external application. Therefore, in this research, the detector data is read directly from the binary files that are transferred from the Traffic Management Center. Figure 7-9 illustrates the structure of the automatic data loading procedure, whose core element is the extraction/conversion module that extracts the traffic data for the detectors at each boundary with the detector list file given by the user interface. The user interface produces the detector list file that is used by the simulation module for ramp metering operations. Currently the extraction/conversion module operates outside of the user interface and will be incorporated into the user interface in the later phase with the completion of the traffic database.

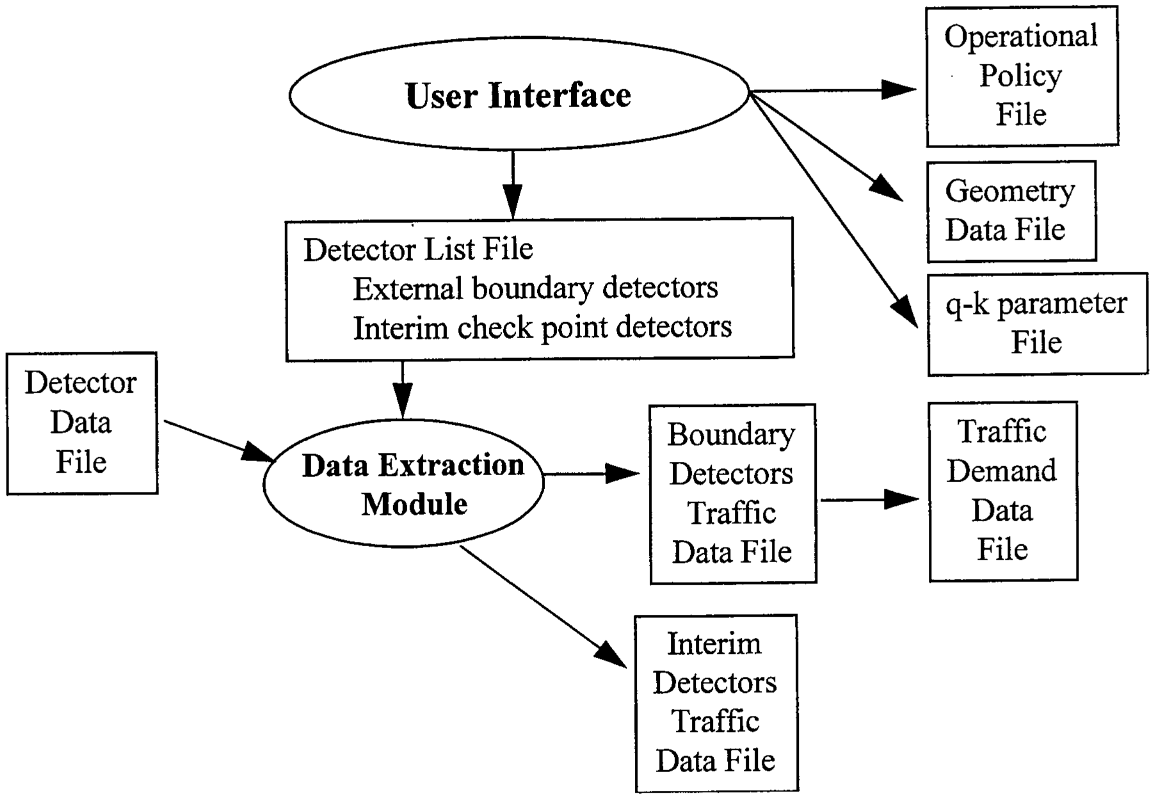


Figure 7-9 Structure of the automatic demand data loading procedure

VIII PRELIMINARY STUDY FOR ESTIMATION OF FLOW-DENSITY RELATIONSHIP

VIII. 1 Flow-density relationship in macroscopic simulation

The macroscopic modeling approach adopted in Kronos requires the flow-density (q-k) relationships for each segment in a given freeway as the user-specified input parameter, whose values substantially affect the simulation results. Figure 8-1 illustrates the general form of a q-k relationship currently used in Kronos. The q-k curve consists of three functional relationships depending on the flow region, i.e.,

$q = a_0 k \; : \; 0 < k <= k_a$

$q = a_1 k^2 + b_1 k + c_1 \; : \; k_a < k <= k_{cr}$

$q = a_2 k^2 + b_2 k + c_2 \; : \; k_{cr} < k <= k_{jam}$

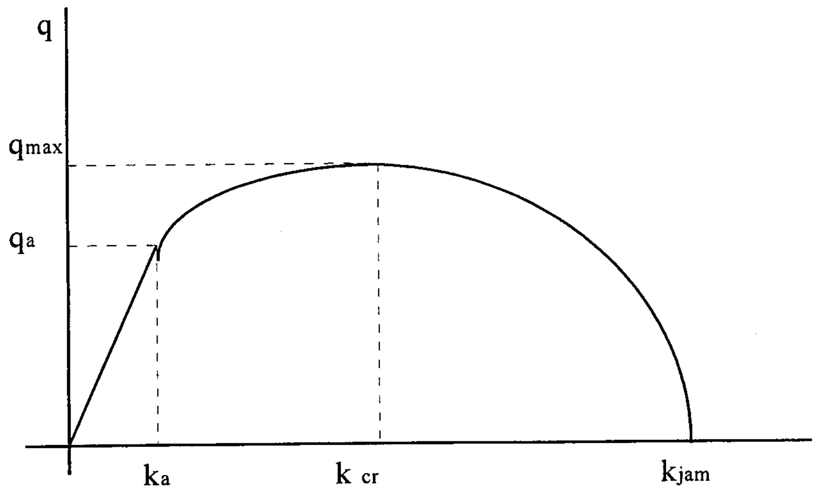


Figure 8-1 General form of flow-density relationship adopted in Kronos

where, a_i, b_i, c_i = parameters for each functional relationship,

k_{cr} = critical density, k_{jam} = jam density,

q_{max} = maximum flow,

q_a, k_a = flow and desity values for the upper limit of the linear q-k relationship.

The parameters in the above q-k relationship can be mathematically identified with the given values of {[(qa, ka), (qmax, kcr), (kjam)]} and Kronos requires user to input those values for each segment. Currently those four key parameter values are determined through a trial-and-error procedure for a given section of a freeway by comparing the simulation results with the data obtained from the loop detectors. This manual calibration procedure has been a time-consuming process largely dependent upon the expertise of the traffic engineers.

VIII. 2 Development of new flow-density curve data structure

Unlike the previous versions limited by the 640KB memory boundary of MS-DOS, the new simulation module developed in this research adopts a dynamic memory allocation scheme and makes a full use of the available memory in a given computer. One of the major benefits from this improvement is the ability to assign a flow-density (q-k) curve for each dx in all segments on the entire stretch of a given freeway section. Each q-k curve information is stored in the q-k structure defined in the simulation module. Each q-k structure variable requires 52 bytes of memory. A dx may need more than one q-k curve structure depending on the flow configuration. For example, a dx that has two separate flows needs three q-k curves; one for mainline, one for either right or left-most lane and one for the entire flow. Enormous memory is required to assign q-k curves for each dx using static memory allocation. Since many dxs have common q-k curves, this process wastes lots of memory. By using dynamic memory allocation, on the other hand, we can optimize the memory allocated to these q-k curves. Using this method, we can eliminate the need for redundancy. In this technique, instead of allocating one q-k curve (52 bytes) for each flow region of a dx, we have a pointer variable (4 bytes) pointing to the corresponding q-k curve in the q-k database. The q-k database consists of all the different q-k curves that are used for the freeway network that is being simulated. This is dynamically built before the beginning of simulation, as explained in a later section. Thus using this method, a q-k curve can be shared by different dxs, thereby, eliminating the redundancy in memory allocation. Figure 8-2 illustrates the q-k database concept.

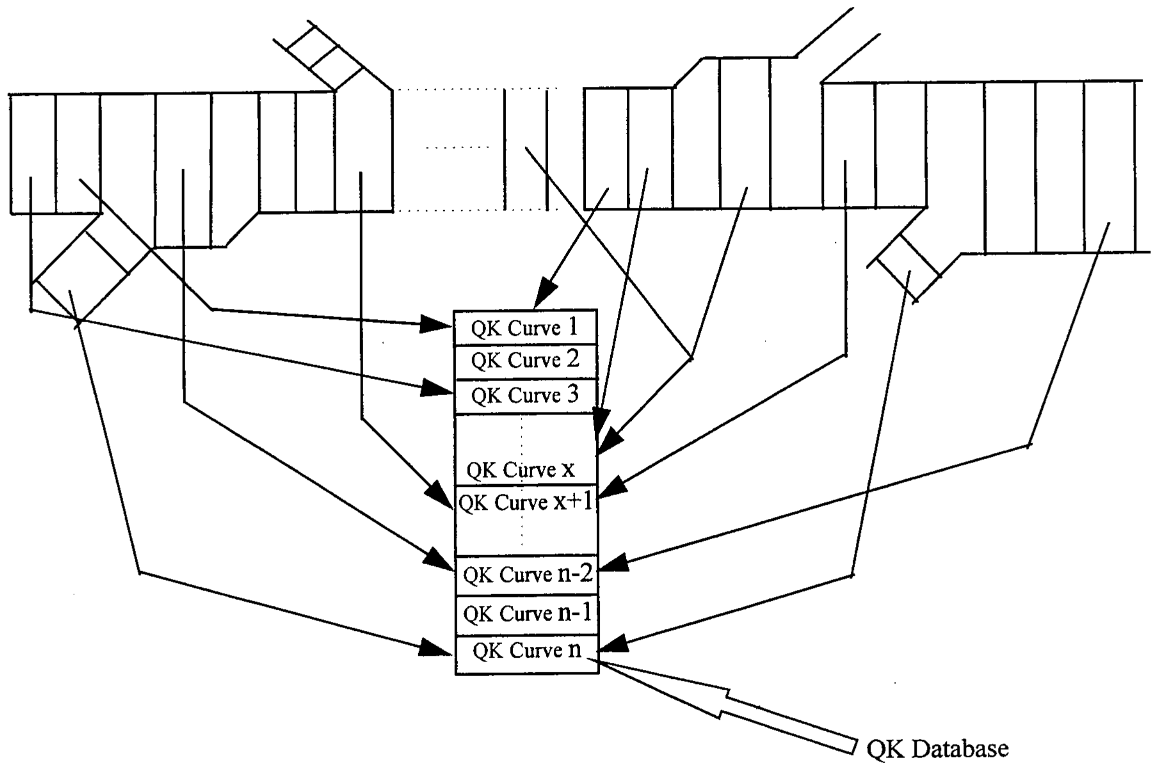


Figure 8-2 Initialization Procedure for QK Curve for each DX on Freeway

Building q-k Database:

The q-k database is implemented in the form of a linked list. This list is empty at the beginning of simulation. In KRONOS, a q-k curve for a flow region is computed for all the lanes in that region. That is, if a region has 'n' lanes, and each lane has a capacity of 'c', then q-k for that region is computed for a total capacity of 'nc'. In KRONOS 9.0, each freeway has global q-k curve parameters, which represent a one lane q-k. These parameters are used as default for any segment on this freeway, unless different q-k parameters are specified. The building of q-k database starts with allocating q-k curves for these global parameters. At the end of this step, we have 1-lane to 8-lane global q-k curves. Since, these q-k curves may be used by many segments, they are kept at the beginning of the list so that time required to scan through the list to find a matching q-k curve is minimized. After this step, for each segment on the freeway, we check to see if gradient is applied at its upstream and downstream ends. If gradient is not applied, all dxs of this segment can point to the same q-k curve. If gradient is applied, those dxs which have the gradient applied, will have different q-k curves. With these parameters, we scan through the list until a matching q-k is found. If a matching q-k curve is not found, we create another q-k curve with the current dx's parameters and append to the existing list. At the end of this phase, we will have all the required q-k curves for this freeway stored in the q-k database.

The new methodology of q-k initialization based on the database concept has offered great flexibility to the program and the modeling. For example,

- Each dx has its own q-k curve, the boundary q-k curve between two segments need not be computed at each DT and hence substantially reducing the performance overhead.
- In KRONOS 8, the capacity gradient was only applied for dxs with one flow. This has restricted the segment regions over which gradient is applied and also the minimum length requirements. This limitation has been eliminated and the capacity gradient method can now be applied between a one-flow region and a multi-flow region. This means that a gradient can be applied at any segment boundary, including ramp-to-ramp, ramp-to-mainline, which has greatly helped in the network freeway modeling by allowing the boundaries of the freeways can have different capacities.

- This also helped to separate the q-k curves of HOV region from the rest of the mainline.
- each dx has its own q-k curve, the program is much simpler. Each dx need not be checked to see whether it is a gradient dx or not.

VIII.3 A framework for an automatic calibration for q-k relationships for a given freeway

Determining the best set of flow-density relationships for a given section of a freeway has been one of the most difficult problems in macroscopic simulation. In this research, a framework for an automatic calibration of the q-k relationships for a freeway section with a given set of real traffic data is developed. Figure 8-3 shows the structure of the q-k calibration process consisting of several major modules. The process starts with the simulation module using initial set of q-k curves or user defined Q-K curves. The simulation results are used by the error estimation module to compare with measurements from field data. The overall error is used to determine if further improvement is needed. If the total error rate is small enough, then the calibration process stops. Otherwise it goes to the q-k calibration module, which finds a new set of q-k curves and the input module creates a new set of q-k parameter file for the next iteration of simulation-comparison-adjustment process. The implementation of each individual module in the framework will be performed in the subsequent phase of this research.

Data extraction module

The data extraction module extracts the traffic data for the detectors specified by user from the raw detector data file provided by the Traffic Management Center. The user interface produces the detector list file from the information given by user. Two traffic data files are created: One for the external boundary traffic data to be used as the demand data for a given freeway section and the other for the interim detectors that will be used as the check points to determine the simulation error.

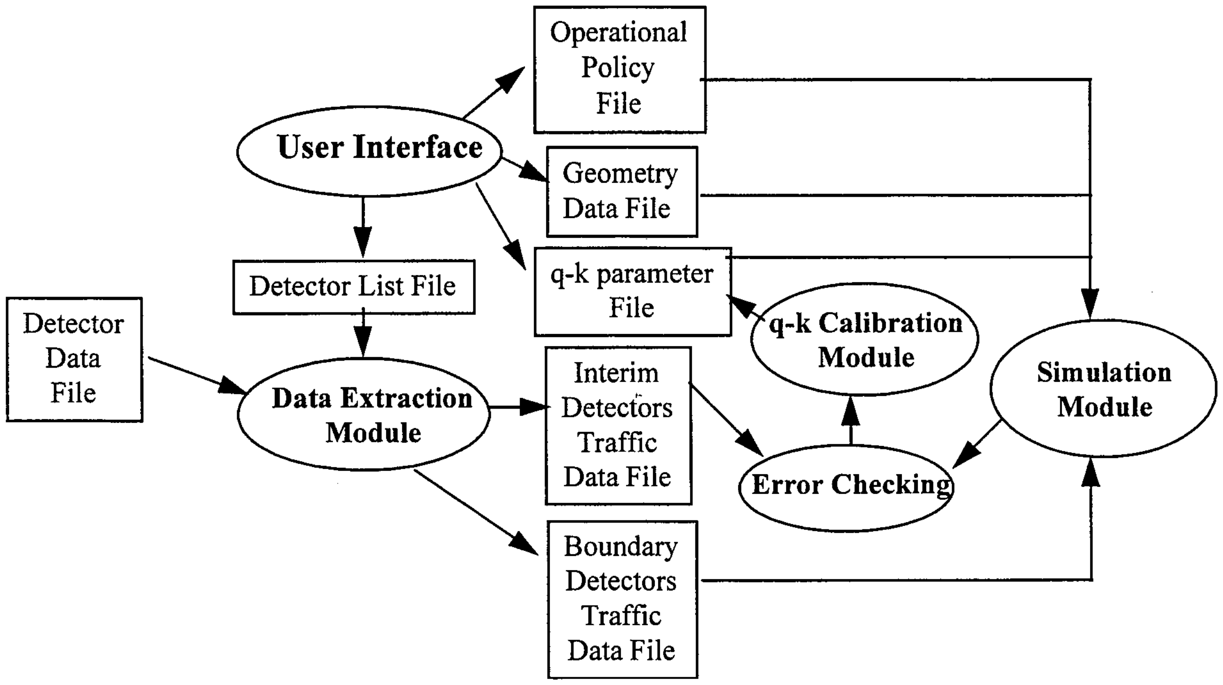


Figure 8-3 Framework for the automatic calibration of flow-density relationships

Error checking module

The error checking module calculates the estimation error by comparing the simulation results at the check points with the real traffic data collected from the same locations. The following error indices are to be used to determine if further adjustment of q-k relationships is needed.

Mean Percentage Difference (MPD) :

$$\text{MPD} = \Sigma\, [\, 100 * (\text{Measured } i - \text{Estimated } i) / \text{Estimated } i\,] / N$$

Mean Absolute Error (MAE) :

$$\text{MAE} = \Sigma\, [\text{Measured } i - \text{Estimated } i\,] / N$$

where, i = check point,

N = number of measured points.

q-k relationship calibration module

Developing an optimal calibration module that can determine the best set of q-k relationships for a given section of a freeway has been one of the most challenging tasks to traffic engineers. In this research, an optimal calibration process for the flow-density relationship of each segment using a least square output error method is developed. It is considered that the q_{max} in the q-k curve is the control variable to be optimized and the initial q-k relationship for each segment can be adjusted with the optimized q_{max} value. Further, the q_{max} values for the segments that do not have detector stations are determined by interpolating adjacent q_{max} values. The complex algorithm, a nonlinear programming search routine developed by Box, has been selected as the optimization procedure. The complex algorithm uses a sequential search technique without requiring the calculation of derivatives and it has been proven to be effective in solving problems with nonlinear objective functions subject to nonlinear inequality constraints(Box, 1975). Using this algorithm, the optimal calibration problem can be formulated as follows;

Find $q_{max,j}$ for each segment j

Minimizing $C(q_{max,j}) = 100\ (\ 0.5\ Cq + 0.5\ Co\ )$: Performance criterion

where, $Cq = \Sigma\Sigma\ (\ |\ q_{i,j} - q(k_{i,j})\ |\ /\ q_{i,j}\ )\ /\ N$: Mean absolute error for volume

$Co = \Sigma\Sigma\ (\ |\ o_{i,j} - o(k_{i,j})\ |\ /\ o_{i,j}\ )\ /\ N$: Mean absolute error for occupancy

where, $q_{i,j}$, $o_{i,j}$ = measured flow rate and occupancy values at time i and from location j.

q(k) and o(k) are estimated flow and occupancy values by simulation.

N = Total number of time sequence, i, * Number of segments, j

Overview of the complex algorithm

The complex algorithm finds Q minimizing the nonlinear performance function

$$C(Q) = 100\ (\ 0.5\ Cq + 0.5\ Co\ )$$

subject to inequality constraints :

$$Lj <= q_{max,j} <= Hj$$

where : Q = a vector for $[q_{max,1},\ q_{max,2},\ ,\ ,\ q_{max,M}]$

i = 1, 2, ... N (Time sequence of simulation) j = 1, 2, ... M (Number of segments)

Lj , Hj : lower and upper constraints of parameters, $q_{max,j}$

Cq , Co : functions of simulated density ki,j

i). For each control variable in the vector Q, generate a starting point based on random numbers and constraints:

$q_{max,j} = Lj + R (Hj - Lj)$ Where R is a random number of 0 and 1.

ii). Check if the selected points satisfy the constraints. If any constraints are violated by a value of d which is defined as 0.00001, the point is moved to a small distance inside the violated limit.

iii). The objective function is evaluated at each point. The point gives the lowest function value is moved to a location at a distance from the centroid of all remaining points.

$Q(n+1) = a * [Qc - Q(n)] + Qc$

where, n : iteration steps

a : reflection factor (a value of 1.3 is recommended)

$Qc = [\Sigma(Q(n) - Q(n-1)] / (\text{Total number of points} - 1)$

iv). If the new point gives lowest function value on next iteration step, it is moved to the location with one half the distance to the centroid.

$Q(n+1) = 0.5 [Qc - Q(n)]$

v). Check if the objective function values at each point are within predefined units, δ, for a certain number of consecutive iterations, η. If that is the case then convergence is reached and program stops. Otherwise it goes back to step 1 for another iteration. The value of convergence parameter δ and η are defined as 0.1 and 5.

Determination of $q_{max,j}$ for the segment without detector measurements

Currently, the loop detectors in the Twin Cities' metro freeways are installed approximately every half a mile. For those segments that do not have the detector measurements, the q_{max} values are determined by interpolating the adjacent q_{max}'s determined by the optimization process with the measurements for each iteration, i.e.,

$$q_{max,j} = q_{max,j-1} + (q_{max,j+1} - q_{max,j-1})*l_j/(l_j+l_{j+1})$$

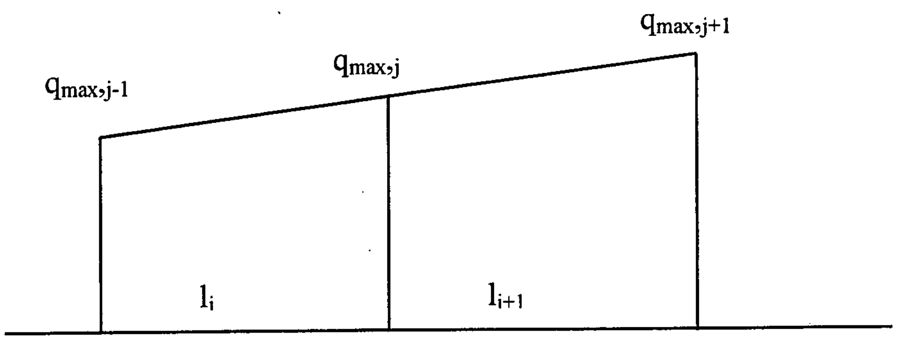


Figure 8-4 Interpolation of $q_{max,j}$

IX. CONCLUSIONS AND FUTURE RESEARCH NEEDS

Performing effective traffic management and improving operational environment requires the ability to assess the effectiveness of various operational and design alternatives prior to implementation. This report summarized the final results of the research effort to develop a freeway traffic simulator that can be applicable in evaluating freeway operational strategies, such as traffic-responsive ramp metering and HOV lanes. First, an efficient software data structure was developed by adopting a dynamic memory allocation scheme to use the available memory as efficiently as possible. The existing macroscopic, segment-based modeling structure was also modified and new types of pipeline segments were developed to facilitate detection modeling and further model enhancements. While this work was not included in the original work plan, the inefficiency in the previous modeling structure, where the segment types were redefined internally to confirm the requirements of the simulation module, was discovered during the process of developing the traffic-responsive ramp metering module. To prevent excessive complexity in modeling detectors and to reduce the possibility of the potential error because of the internal type conversion, the new definition of the freeway segment types was introduced and the new simulation procedure for each new segment type was developed. Due to the additional work involved in this modeling structure modification, the testing of the ramp metering module with the real traffic data will be conducted later jointly with the MnDOT traffic engineers. Based on the new segment-based modeling structure, a new simulation module to handle HOV lane traffic flows was developed and the simulation procedure to treat an exclusive HOV lane was extended to handle a network of freeways. Further, a new module to emulate the traffic-responsive ramp metering algorithm implemented by the Traffic Management Center since 1980's has been developed and incorporated into the simulation module. To facilitate the data input process for the expanded simulation features, the enhancement of the existing DOS-based user interface was initially tried. However, it was soon found out that it would be very difficult to modify the existing user interface because of the limitations in the DOS-based programming environment. As a result, a new Windows-based user interface was developed using the Delphi software as the development tool kit. The new user interface was developed as a full Windows-based application and most of the data input process could be done without exiting the main menu screen. The demand data conversion module that extracts the traffic data from the raw detector data file and converts them into the traffic demand data for simulation was also developed as the first step to integrate simulation and data management. The preliminary study to automatically estimate the flow-density relationships for a given section of a freeway with real traffic data was also performed and a framework for an optimal calibration process for the flow-density relationships was developed using a non-linear optimization routine.

A number of further research needs have been identified. First, the software developed from this research needs to be debugged and tested with various realistic cases. The budget and time limitations in this research did not allow to perform an extensive testing and debugging of the software. Further, the screen design of the new user interface can be improved with the feedback from the traffic engineers. Secondly, the traffic models developed and incorporated into the simulation module need to be continuously tested and enhanced with the real traffic data from various traffic and weather conditions. The flow-density relationships of the various segment types under the different traffic and weather conditions need to be studied and a base set of default relationships can be developed. Finally, an efficient traffic database that can store and manage the traffic data from the existing detectors needs to be developed, so that the user interface can directly extract the traffic data from the database and create the demand data necessary for simulation. Developing an integrated system where evaluation of alternative operational strategies can be performed with efficient interaction between simulation and database modules is of critical importance in improving traffic operations and management under dynamically changing traffic environment.

BIBLIOGRAPHY

Michalopoulos, Panos and Kwon, Eil et. al. Enhancements of the KRONOS simulation package for geometric design, planning, operations and traffic management in freeway networks/corridors, Phase I, Final Report prepared for Minnesota Department of Transportation, University of Minnesota, March 1993.

Michalopoulos, Panos and Kwon, Eil and Kang, J. Enhancements and field testing of a dynamic freeway simulation program. Transportation Research Record 1320, National Research Council, pp. 203-215, 1991.

Kwon, Eil, Michalopoulos, Panos, et. al. Enhancements of the KRONOS simulation package for geometric design, planning, operations and traffic management in freeway networks/corridors, Phase II, Final Report prepared for Minnesota Department of Transportation, University of Minnesota, 1995.

May, Adolf. Traffic flow fundamentals, Prentice Hall, 1990.

Lax, P. Weak solution of non-linear hyperbolic equation and their numerical computations, Comm. Pure. Appl. Math. 7, pp. 159-193, 1954.

Lau, Rich, Minnesota Traffic Responsive ramp metering algorithm, Internal documentation, 1997.

Borland International Inc., Boralnd Delphi for Windows 95 and Windows NT, version 2.0, Borland International Inc. 1996.

Cremer, M., and Papageorgiou, M., "Parameter Identification for a Traffic Flow Model" , Brief paper, Automatica.

J. L. Kuester, M.H. Joe, "Optimization Techniques with Fortran". McGraw-Hill Book Company

M.J. Box "A new method of constrained optimization and a comparison with other methods", Comput. J., 1975.